

Manfred Reichert, Stefan Strecker, Klaus Turowski (Eds.)

## **EMISA 2007**

### **Enterprise Modelling and Information Systems Architectures - Concepts and Applications -**

**Proceedings of the 2<sup>nd</sup> International Workshop on Enterprise Modelling and Information Systems Architectures**

**St. Goar, Germany  
October 8-9, 2007**

Gesellschaft für Informatik 2007

## **Lecture Notes in Informatics (LNI) - Proceedings**

Series of the Gesellschaft für Informatik (GI)

Volume P-119

ISBN 978-3-88579-213-0

ISSN 1617-5468

### **Volume Editors**

Prof. Dr. Manfred Reichert

University of Twente

Faculty of Electrical Engineering, Mathematics & Computer Science

P.O. Box 217, 7500 AE Enschede, The Netherlands

Email: m.u.reichert@cs.utwente.nl

Dr. Stefan Strecker

Universität Duisburg-Essen

Institut für Informatik und Wirtschaftsinformatik

Universitätsstr. 9, 45141 Essen, Germany

Email: stefan.strecker@uni-duisburg-essen.de

Prof. Dr. Klaus Turowski

Universität Augsburg

Wirtschaftsinformatik und Systems Engineering

Universitätsstr. 16, 86159 Augsburg, Germany

Email: klaus.turowski@wiwi.uni-augsburg.de

### **Series Editorial Board**

Heinrich C. Mayr, Universität Klagenfurt, Austria (Chairman, mayr@ifit.uni-klu.ac.at)

Jörg Becker, Universität Münster, Germany

Ulrich Furbach, Universität Koblenz, Germany

Axel Lehmann, Universität der Bundeswehr München, Germany

Peter Liggesmeyer, TU Kaiserslautern und Fraunhofer IESE, Germany

Ernst W. Mayr, Technische Universität München, Germany

Heinrich Müller, Universität Dortmund, Germany

Heinrich Reinermann, Hochschule für Verwaltungswissenschaften Speyer, Germany

Karl-Heinz Rödiger, Universität Bremen, Germany

Sigrid Schubert, Universität Siegen, Germany

### **Dissertations**

Dorothea Wagner, Universität Karlsruhe, Germany

### **Seminars**

Reinhard Wilhelm, Universität des Saarlandes, Germany

© Gesellschaft für Informatik, Bonn 2007

printed by Köllen Druck+Verlag GmbH, Bonn

## Preface

Modern organizations recognize the need for a close alignment of their business and IT. Achieving such an alignment recommends the co-design of the organization and its information systems considering in particular the corporate strategy, business processes, and the information systems that support them. In this respect, two essential challenges pertain to reducing the inherent complexity of co-design, and to overcoming the notorious cultural chasm between business people and IT professionals.

Conceptual models of the enterprise as well as information systems architectures represent important means to deal with these challenges. Enterprise models integrate conceptual models of information systems and models of the surrounding action systems such as business process models and, hence, take into account technical, organisational, as well as economic aspects of the organization. Information systems architectures provide ‘blueprints’ for the design and implementation of software systems and complement enterprise models in the co-design of the organization and its information systems. Both serve as a medium to foster communication and cooperation between various stakeholders in the firm. At the same time, research on enterprise models and information systems architectures recommends the cooperation of fields such as Information Systems, Business Informatics, and Computer Science.

The 2<sup>nd</sup> International Workshop on “Enterprise Modelling and Information Systems Architectures – Concepts and Applications” (EMISA’07) addresses all aspects relevant for enterprise modelling as well as for designing enterprise architectures in general and information systems architectures in particular. It is jointly organized by the GI Special Interest Group on Modelling Business Information Systems (GI-SIG MoBIS) and the GI Special Interest Group on Design Methods for Information Systems (GI-SIG EMISA).

These proceedings feature a selection of 15 high quality contributions from academia and practice on enterprise architecture models, business processes management, information systems engineering, and other important issues in enterprise modelling and information systems architectures. We received 39 submissions which were all thoroughly reviewed by at least two selected experts of the program committee. Fifteen contributions were selected for presentation at the workshop and for publication in these proceedings.

We would like to thank the members of the program committee and the reviewers for their efforts in selecting the papers. They helped us to compile a high-quality technical program. We would like to acknowledge the splendid support of the local organization. We also thank Mathias Weske as keynote speaker. Special thanks go to Ulrich Frank and Peter Rittgen for their assistance with organizing EMISA’07. We hope you will find the papers in this volume interesting and stimulating.

October 2007

Manfred Reichert, Stefan Strecker, and Klaus Turowski (Eds.)



## **Organisation**

Dr. Peter Rittgen  
Senior Lecturer  
School of Business and Informatics  
University College of Borås  
S-501 90 Borås, Sweden

Dr. Stefan Strecker  
Assistant Professor  
Information Systems and Enterprise  
Modelling Research Group  
University Duisburg-Essen  
Universitaetsstr. 9,  
45141 Essen, Germany

The workshop is jointly organized by the GI Special Interest Group on Modelling Business Information Systems (GI-SIG MobIS) and the GI Special Interest Group on Design Methods for Information Systems (GI-SIG EMISA):

### **GI-SIG MobIS**

Conceptual Modelling is pivotal for analysing and designing information systems that are in line with a company's long term strategy and that efficiently support its core business processes. The Special Interest Group on Modelling Business Information Systems (SIG MobIS) within the German Informatics Society (GI) aims at providing a forum for exchanging ideas and solutions on modelling research within Information Systems - both for researchers at universities and for experts in industry.

### **GI-SIG EMISA**

The GI Special Interest Group on Design Methods for Information Systems provides a forum for researchers from various disciplines who develop and apply methods to support the analysis and design of information systems.

## Program Committee

Manfred Reichert (University of Twente)  
Co-Chair  
m.u.reichert [at] utwente.nl

Klaus Turowski (University of Augsburg)  
Co-Chair  
klaus.turowski [at] wiwi.uni-augsburg.de

W. Abramowicz (Poznan University)  
P. Ågerfalk (University of Limerick)  
A. Albani (University of Augsburg)  
C. Atkinson (University of Mannheim)  
P. Bøgh Andersen (Aarhus University)  
L. Bækgaard (Aarhus School of Business)  
J. Becker (University of Münster)  
M. Bertram (Commerzbank Frankfurt)  
J. Desel (KU Eichstätt)  
W. Esswein (TU Dresden)  
M. Favier (Université PMF Grenoble)  
F. Feltz (CREDI Luxembourg)  
U. Frank (University of Duisburg-Essen)  
A. Gadatsch (FH Siegburg-Bonn)  
C. Godart (Université Nancy)  
U. Greiner (SAP Research, Karlsruhe)  
W. Hasselbring (University Oldenburg)  
B. Henderson-Sellers (University of Technology Sydney)  
W.J. van den Heuvel (Tilburg University)  
H. Jasper (TU Freiberg)  
F. Karlsson (Örebro University)  
D. Karagiannis (University of Vienna)  
R. Kaschek (Massey University)  
R. Klischewski (German University Kairo)  
J. Krogstie (University of Trondheim)  
D. Kuropka (HPI Potsdam)

S. Leist (University of Regensburg)  
S.W. Liddle (Brigham Young University)  
M. Lind (University College of Borås)  
K.-W. Müller (MSG Systems, München)  
M. Nüttgens (University of Hamburg)  
A. Oberweis (University of Karlsruhe)  
E. Ortner (TU Darmstadt)  
S. Overhage (University of Augsburg)  
H.J. Paul (Institut Arbeit und Technik)  
E. Proper (Radboud University, Nijmegen)  
M. Rebstock (University of Applied Sciences Darmstadt)  
S. Rinderle (University of Ulm)  
P. Rittgen (University College of Borås)  
M. Rosemann (QUT, Brisbane)  
M. Rossi (Helsinki Business School)  
G. Saake (University of Magdeburg)  
G. Sindre (University of Trondheim)  
E. J. Sinz (University of Bamberg)  
S. Strecker (University of Duisburg-Essen)  
J.-P. Tolvanen (University of Jyväskylä)  
G. Vossen (Universität Münster)  
B. Weber (Universität Innsbruck)  
H. Weigand (Tilburg University)  
M. Weske (HPI Potsdam)  
R. Wieringa (University of Twente)  
R. Winter (University of St. Gallen)

## Table of Contents

### Enterprise Architecture Models

<i>A Federated Approach to Enterprise Architecture Model Maintenance</i>	9
Ronny Fischer, Stephan Aier, Robert Winter	
<i>EA Model as Central Part of the Transformation Into a More Flexible and Powerful Organisation</i>	23
Stefan Gerber, Uwe Meyer, Claus Richert	
<i>Generating Visualizations of Enterprise Architectures using Model Transformations</i>	33
Sabine Buckl, Alexander M. Ernst, Josef Lankes, Christian M. Schweda, André Wittenburg	

### Architecture Principles

<i>Architecture Principles - A Regulative Perspective on Enterprise Architecture</i>	47
Patrick van Bommel, Pieter Buitenhuis, Stijn Hoppenbrouwers, Erik Proper	
<i>Service Oriented Security Architecture</i>	61
Cristian Opincaru, Gabriela Gheorghe	
<i>An Approach to use Executable Models for Testing</i>	75
Michael Soden, Hajo Eichler	

### Business Process Management

<i>Modelling of Cross-Organizational Business Processes</i>	87
Jörg Ziemann, Thomas Matheis, Jörn Freiheit	
<i>Using BPEL as a Workflow Engine for Local Enterprise Applications</i>	101
Nicolas Biri, Pascal Bauler, Fernand Feltz, Nicolas Médoc, Céline Thomase	
<i>BPMN-Q: A Language to Query Business Processes</i>	115
Ahmed Awad	

## **Information Systems Engineering**

- A Practical Approach to Ontology-based Software Engineering* 129  
Andrej Bachmann, Wolfgang Hesse, Aaron Ruß, Christian Kop, Heinrich C. Mayr,  
Jürgen Vöhringer
- Viewpoint-based Meta Model Engineering* 143  
Stephan Kurpjuweit, Robert Winter
- Design and Usage of an IT-System for Workplace Management with Ergonomic  
Analysis Under Health Protection Aspects* 163  
Clemens Dubian, Wolfgang May

## **Issues in Modelling**

- On Industrial Use of Requirements Engineering Techniques* 177  
Lars Bækgaard, Jens Bæk Jørgensen, Kristian Bisgaard Lassen
- UML 2 Profiles for Ontology Charts and Diplans - Issues on Metamodeling* 191  
Jose Cordeiro, Kecheng Liu
- Service Modelling: A Hybrid Approach In Decomposed Financial Value Chains* 205  
Falk Kohlmann



# A Federated Approach to Enterprise Architecture Model Maintenance

Ronny Fischer, Stephan Aier, Robert Winter

Institute of Information Management  
University of St. Gallen  
Müller-Friedberg-Strasse 8  
CH-9000 St. Gallen  
{ronny.fischer | stephan.aier | robert.winter}@unisg.ch

**Abstract:** Enterprise architecture is gaining acceptance as an approach to manage change and foster IT/business alignment by (1) propagating strategy and process changes to the software and infrastructure level, by (2) supporting consistent business transformation enabled by technology innovations, and by (3) decoupling business-oriented and technology-oriented architectures. Due to constant change in business as well as in technology, enterprise architecture management is a permanent process rather than a one-time effort. To keep enterprise architecture models up-to-date, a well-engineered maintenance concept including processes, roles and schedules is needed. This paper discusses the shortcomings of existing approaches to enterprise architecture model maintenance, proposes a federated approach, and reports on its implementation at a large financial service provider.

## 1 Introduction

In recent years, companies are exposed to frequent changes in their social and economic environment. In particular, companies are faced with major challenges such as:

1. An increasing complexity of business transactions due to customization of products and services as well as growing globalization with respect to service development, service creation, and service distribution [RWR06; Wa05].
2. An accelerated rate of change in business models due to fierce, international competition [RWR06; Sc05a; Wa05].
3. A growing regulatory framework, which forces companies to prove that they have a firm understanding of their operations and that they comply with applicable regulations [La05; Sc05a] such as Sarbanes-Oxley Act (SOX), Basel II or Solvency II.
4. A growing dependency on information technology which enables completely new products and business processes [Da93; Ve91]. As a consequence thereof, companies are increasingly threatened by technology-related risks.

Companies have to adapt their corporate strategies continuously and have to align corporate structures with strategic goals. Corporate structures comprise organizational structures and processes as well as supporting information systems and technologies. Enterprise architecture (EA) describes the fundamental structure of an enterprise [Og03; Ro94; Sc04; WF06] and supports transformation by offering a holistic perspective of as-is as well as to-be structures and processes [La05].

EA is gaining acceptance as an approach to manage change and foster IT/business alignment by (a) propagating strategy and process changes to the software and infrastructure level, by (b) supporting consistent business transformation enabled by technology innovations, and by (c) decoupling business-oriented and technology-oriented architectures [BS02; RWR06; Ve01; Wa05]. Empirical studies confirm the strategic importance of EA. According to a study conducted in 2005 by the Institute for Enterprise Architecture Developments (IFEAD), 66% of the respondents consider EA as an important element of their strategic governance processes [Sc05b]. Another study conducted in 2006 among Swiss and German companies reveals that 38 from 51 interviewed companies are either currently implementing EA models, or are already using EA models [Bu06]. Besides supporting strategy execution, a large number of other EA application scenarios exist, e. g. business continuity planning, security management, compliance management and sourcing management [Bu06; RB06]. EA is the primary tool for impact assessment and tradeoff analysis in these scenarios.

In summary it can be stated that the main goals of EA are

1. documentation and communication of as-is corporate structures/processes,
2. support for the design of to-be structure/processes, and
3. support for projects that transform as-is into to-be structures/processes.

EA models support these goals by creating more transparency, measurability, and consistency. Consequently, EA models must remain up-to-date and reflect the current state of corporate structures and processes [Ci01]. Hence, EA models need regular maintenance [La05]. This necessitates processes for EA management and communication in general, and in particular a specific organizational design that ensures the completeness and consistency of EA models over time.

Various approaches for managing EA have been developed by academia as well as by practitioners. Documentation of these approaches differs substantially with respect to quantity and formalization. A common problem is a lack of completeness and/or insufficient level of detail. In particular, existing approaches to EA management pay little attention to specifying maintenance procedures for EA models in detail. Given the shortcomings of existing approaches, this paper focuses on the maintenance process and reports on a federated approach to maintain a current-state EA model.

The remainder of this paper is organized as follows: In section 2 we analyze several existing approaches to EA management. Based on this analysis, we specify the research gap. Possible basic strategies for EA maintenance are discussed in section 3. In section 4

we propose a federated approach to EA maintenance. The implementation of this approach at a large financial service provider is presented in section 5. In section 6, conclusions regarding success factors and obstacles for federated EA maintenance are drawn, and an outlook to further research is given.

## 2 State-of-the-Art of Enterprise Architecture Maintenance

A multitude of methods for enterprise architecture management has been developed by academia and practitioners (e. g. [Az05; Az06; BK05; Ci01; Dv01; If99; Og03; SH93; Wa05]). These methods usually distinguish between the following EA management processes: (a) strategic dialogue/architecture visioning, (b) development and maintenance of current-state EA models, (c) development and maintenance of future-state EA models, (d) migration planning, and (e) EA implementation.

Documentation of the aforementioned approaches differs substantially with respect to quantity, level of detail, and formality. Even worse, almost all of these approaches to EA management pay little attention to specifying maintenance procedures for EA model data in detail. In order to substantiate this assessment, we provide an analysis of three popular, comprehensive approaches to EA management on how much they incorporate maintenance aspects. These approaches include the Chief Information Officer Council's "A Practical Guide to Federal Enterprise Architecture" [Ci01], the Open Group's "TOGAF" (The Open Group Architecture Framework Version 8.1 "Enterprise Edition") [Og03], and Wagter's et al. "Dynamic Enterprise Architecture: How to Make It Work" [Wa05].

While [Ci01] and [Wa05] mention an EA maintenance process, EA maintenance activities are not specified in detail, and specific roles/responsibilities are not defined. Although it has to be mentioned, that the Chief Information Officer Council defines a maintenance process for their own reference model [Ci05]. This process may be adapted for maintaining EA models, too. TOGAF [Og03], one of the most widely-used approaches, does not even mention a maintenance process. Other researchers come to the same conclusion. As Jonkers et al. state: "The instruments needed for creating and using enterprise architecture are still in their infancy" [Jo06]. Given the lack of existing approaches, the following research question is addressed in this paper:

*How should an EA maintenance concept be designed to ensure the sustainable and efficient usage of EA as an instrument for strategic change and alignment?*

In a design research approach [He04], this contribution pursues the following design goals:

- Design of operational structures for EA maintenance: Detailed, formal description of a process necessary to maintain EA content.
- Design of organizational structures for EA management: Specification of roles to execute, manage and control all maintenance process activities.

- Integration of operational and organizational structures: Mapping of roles to process activities by means of responsibility charting (i.e. by specification of responsibility, accountability, etc. for each process activity).

### 3 The Challenge of Enterprise Architecture Maintenance

EA is comprised of a large number of business related and IT related artifacts. Popular framework approaches to EA including [Ci99; Og03; Sc99; WF06] propose the following set of EA core artifacts:

- *Strategy specification* (“what” questions): Hierarchy of organizational goals and success factors, product/service model (including partners in value networks), targeted market segments, core competencies, strategic projects, business principles, and dependencies between these artifacts.
- *Organization/process specification* (“how” questions): Specification of structure (organizational unit hierarchy, business location hierarchy, business role hierarchy, dependencies between these artifacts), specification of behavior (business function hierarchy, business process hierarchy including in-puts/outputs, internal and external business services including service levels, performance indicators, service flows), specification of information logistics (business information objects, aggregate information flows), and dependencies between these artifacts (e.g. responsibilities, information requirements).
- *Integration/Application specification* (IT/business alignment questions): Specification of applications and application components, enterprise services, service components and dependencies between these artifacts.
- *Software specification*: Specification of software components (functionality hierarchy, event/message hierarchy), data resources (conceptual, logical and physical data models), and dependencies between these artifacts (e.g. data usage by software components CRUD).
- *Technical infrastructure specification*: Specification of IT components (hardware units, network nodes, etc.) and dependencies between these artifacts.
- *Specification of dependencies between layers*, e.g. organizational goals/success factors vs. process metrics, products/services vs. process deliverables, organizational units vs. applications (“ownership”), activities vs. applications, activities/business processes/information requirements vs. enterprise services (“orchestration”), applications/enterprise services vs. conceptual data entity types, and applications/enterprise services vs. software components (“composition”).

Most of the EA artifact classes can be modeled as aggregation hierarchies, i.e. can be represented on various levels of aggregation. It is obvious that the complexity of a medium or large corporation (or government agency) cannot be covered by one single EA

model. In real life, several models for different parts of the enterprise might be maintained, and/or EA will co-exist with other, more specialized architectures that cover a subset of those artifacts [Be05; WF06]. EA comprises only aggregate artifacts and their relationships within and across all layers (cf. Fig. 1).

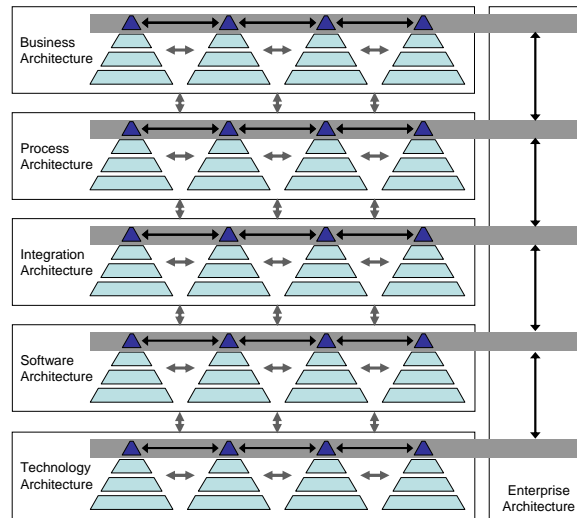


Fig. 1: Enterprise architecture as cross-layer view of aggregate artifacts [WF06]

We agree with other researchers, that EA modeling should focus on consolidating models, modeling techniques and tools already existing in a company and integrating these at an appropriate level of abstraction [DL04]. Hence useful interfaces between EA and specialized architectures have to be specified and maintenance processes have to be established. According to [WF06], appropriate interfaces to at least the following specialized architectures are needed:

- product/service architecture (managed e.g. using a product management tool),
- metrics architecture (managed e.g. using a performance management tool),
- process architecture (managed e.g. using a process modeling tool and workflow management systems),
- information/data architecture (managed e.g. using a data modeling tool and database management systems),
- software architecture (managed e.g. using a software design tool and a configuration management tool), and
- technology architecture (managed e.g. using a computer system management tool).

Basically, two strategies for maintaining architectural data exist [Mü06]:

1. establishing a holistic EA model, or
2. implementing a federated EA model.

A holistic EA model means that there is only a single model comprising all artifact classes necessary to describe EA. Models from specialized architectures are submitted to the EA team. The EA team interprets these models and remodels them using the components specified in the EA meta-model.

A federated EA model means that existing models (that originate from specialized architectures) are used. These models are linked to the EA model by meta-model integration. Two possibilities for model data management exist in this context: (a) Either retrieving model data on the fly when generating EA reports or (b) storing a copy (of the relevant subset) of model data from specialized architectures in the EA repository and periodically updating these data.

The latter strategy was chosen for the approach we propose in the next section. A federated bottom-up approach supported by a common set of rules requires less management effort (especially if specialized models change), provides up-to-date data, yields a higher acceptance of the resulting EA models, and avoids misinterpretation of specialized models during remodeling [Br03; Mi79; PL77].

## **4 A Federated Approach to Enterprise Architecture Maintenance**

To address the challenge of keeping EA models up-to-date, we propose a federated approach. In this approach, the EA repository is designated to store a copy of model data from specialized architectures relevant for EA purposes. Formerly independent models from specialized architectures are linked to the EA repository.

### **4.1 Maintenance Concept**

We suggest that an EA model should – wherever possible – use data from existing specialized architectures to keep modeling efforts low. This necessitates the implementation of interfaces to source systems storing model data of specialized architectures into the EA repository and the establishment of a formal data maintenance process (ref. section 4.2) for each data source. To ensure data quality, we propose the concept of data delivery contracts. A data delivery contract includes a definition of the interface to the source system, descriptions of model data from the specialized architecture to be stored in the EA repository, transformation rules and a maintenance schedule. Data maintenance processes are executed in regular intervals. Special events however, may trigger additional maintenance cycles. Before model data from specialized architectures are stored in the EA repository, consistency checks are performed.

## 4.2 Maintenance Process

To derive the maintenance process, we followed the process design method Promet BPR [Ba96; Im97; Ös95]. In this context, the respective specialized architecture model to be updated defines the core business object around which the processes are built [Ös95]. In order to promote a comprehensive specification of maintenance process tasks, we used the generic activities proposed in [Ma03; Ma99].

We distinguish between a periodic and a non-periodic maintenance cycle. A periodic maintenance-cycle is initiated by the EA team based on the maintenance schedule defined in the data delivery contract. The EA team informs the respective data owner to provide the model data defined in the data delivery contract.

Non-periodic maintenance cycles may be triggered by the EA team as well as the respective data owner. These cycles are initiated e.g. if models of specialized architectures have changed significantly due to project work. At the end of the project the respective data owner informs the EA team about the changes. The EA team then decides whether or not a non-periodic maintenance cycle for this data source is necessary.

Apart from the triggering event of the maintenance cycle (activity 1), further operational sequences are identical for periodic and non-periodic maintenance cycles. Fig. 2 depicts the complete process sequence. Process activities are numbered. Swim lanes denote accountabilities of the roles involved in process execution (for details cf. section 3.3).

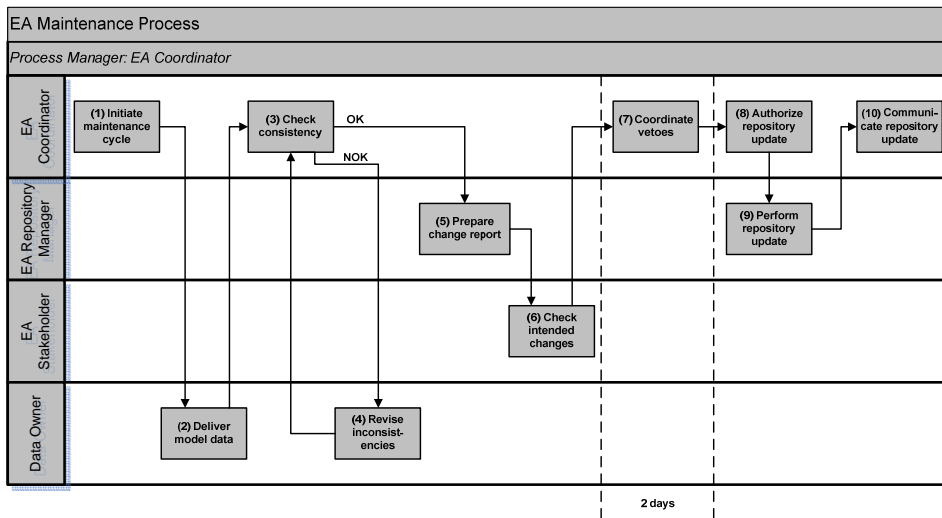


Fig. 2: EA Maintenance Process

First, on request by the EA team, the respective data owner delivers updated model data of its specialized architecture as specified in the data delivery contract (activity 2). The data owner is responsible for providing model data in the correct data format. In most cases data will be delivered as an XML or CSV file, as most EA tool vendors provide

technically mature concepts for fully automated data transfer. The EA team subsequently performs consistency checks with the model data from specialized architectures (activity 3). In case of inconsistencies the data owner gets informed and is requested to revise the data set (activity 4)). After the revision, the data owner resubmits the data set to the EA team. The EA team again checks the revised data set and decides whether another revision cycle is necessary or not.

If the data set has eventually passed the consistency check, the EA team prepares a report which contains all intended changes to EA models (activity 5). The report is derived through a comparison between data currently stored in the EA repository and the updated dataset from the respective specialized architecture model. This report is sent to all affected EA stakeholders (i.e. to all departments which have subscribed to EA reports using those data intended to change). The affected EA stakeholders evaluate the intended changes (activity 6). If a stakeholder enters an objection, the EA team must initiate a process of coordination involving the stakeholder who vetoed, the data owner, and – if necessary – other EA stakeholders who might be affected (activity 7).

If all issues are resolved (i.e. if all stakeholders have finally approved the intended changes), the EA coordinator authorizes the EA repository manager to load the updated data into the EA repository and build a new version of the current-state EA (activity 8). Finally, after loading the updated data into the EA repository (activity 9), the availability of a new release of the current-state EA is communicated to all EA stakeholders (e.g. via e-mail, activity 10).

### **4.3 Roles**

In this section we describe the roles involved in EA maintenance activities. These roles are derived from the organizational units involved in the model update process. Regarding the activities which have to be performed by the EA team, we differentiate between a technology orientated management role (EA repository manager) and a business orientated management role (EA coordinator) because the required qualification profiles are widely different. In addition, we define the roles of EA stakeholders and data owners of specialized architectures.

Being not involved in maintenance activities, the chief enterprise architect is informed about repository updates on a regular basis. The maintenance process is managed by the EA coordinator. The EA coordinator is a member of the EA team. He or she reports to the chief architect. His or her main responsibilities include EA meta-model enhancement, specification of interfaces to specialized architectures, maintenance of EA repository data, and design of EA reports.

The EA repository manager is responsible for all technical issues related to the EA repository. These include user administration, software updates, data backup, and particularly loading updated model data from specialized architectures into the repository. He or she is a member of the EA team.



EA stakeholders are business and IT units using EA to facilitate the understanding of multi-layer dependencies within different application scenarios (e. g. strategy execution, business continuity planning, and security management). Each business or IT unit representing an EA stakeholder names a contact person. The contact person ensures fast and effective communication between the EA team and the respective organizational unit.

For every specialized architecture, a data owner should be defined. On request by the EA team, the data owner provides model data to keep the EA repository up-to-date. Furthermore he or she assists the EA team in specifying and maintaining the interface between the EA repository and the specialized architecture repository or modeling tool.

Table 1: RACI matrix for EA maintenance process

Activities	Roles				
	Chief Enterprise Architect	EA Coordinator	EA Repository Manager	EA Stakeholder	Data Owner
(1) Initiate maintenance cycle		A, R	I		R
(2) Deliver model data from specialized architecture		I			A, R
(3) Check data consistency		A	R		I
(4) Revise inconsistencies		C	I		A, R
(5) Prepare change report & notify affected stakeholders		I	A, R	I	
(6) Check intended changes		I		A, R	
(7) Coordinate vetoes		A, R	I	C	C
(8) Authorize repository update		A, R	I		
(9) Perform repository update		I	A, R		
(10) Communicate repository update	I	A, R	I	I	I
<b>Responsible</b>	Position working on the activity				
<b>Accountable</b>	Position with yes/no authority				
<b>Consult</b>	Position involved prior to decision or action				
<b>Inform</b>	Position that needs to know of the decision or action				

Table 1 presents the RACI matrix [SE07] used to describe the responsibilities of the roles involved in the EA maintenance process in detail. It is especially useful in clarifying roles and responsibilities in cross functional/cross departmental processes such as the one at hand. The RACI matrix breaks maintenance tasks down to four responsibility types that are then assigned to the different roles involved in the maintenance of the current-state EA.

## **5 Implementation at a Large Financial Service Provider**

This section reports on the evaluation of our federated approach to EA maintenance. We use the case of a large financial service provider which implemented our approach. Unlike many other organizations, IT/business alignment has not been the major driver for EA efforts in this company. Instead, EA aims at supporting strategy implementation, in particular at supporting the project selection/project portfolio planning process. In addition, EA is regarded as foundation of business continuity planning, service management and security management.

The financial service provider's EA program was initiated in 2005 because an aggregate, enterprise-wide view of important entities and dependencies did not exist. The program is ongoing and aims at establishing EA as a service to business and IT units. The project we report on has been carried out in 2006 and belongs to a comprehensive EA program. It was started because past approaches to solve the problem of managing the intertwined dependencies of EA artifacts were expensive, since they required scarce experienced architects, time consuming, since the required data were not at hand, frequently incomplete, since the effort to document every aspect was not justifiable, and often out of date since the ongoing expense of maintaining this information was too high [see also GKC06].

In order to address the challenge of keeping EA data up-to-date, the financial service provider decided to pursue a federated approach. In this approach the responsibility for maintaining artifact descriptions is delegated to the team that is responsible for this artifact class. A self-developed EA repository (based on a relational database) has been implemented to store a copy of model data from those specialized architectures (Fig. 3) which are relevant for EA purposes.

If needed for analyses, formerly independent models from different specialized architectures have been linked. Interrelating models was accomplished by the EA team together with the respective data owners of the underlying models. Relationship ownership was eventually assigned to one of the participating data owners for further maintenance. For each data source, a maintenance process similar to the one described in section 4 has been established and the necessary roles have been implemented in the organizational structure.

Model data transfer from specialized architectures to the EA repository is primarily accomplished by means of CSV files. A fully automated data transfer is considered as a future option. While efforts for an automation of repository updates will keep within

reasonable limits for clearly structured data with well defined intersubjectively comprehensible semantics like, e.g. hardware inventory data, automation will be more complex for e.g. business process models exported from process modeling tools. Therefore the implementation of automated repository updates has to be decided on a case by case basis.

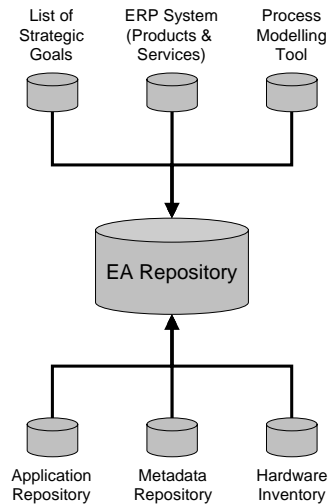


Fig. 3: Primary data sources for EA content

After the first domain-specific repository has been connected to the EA repository in February 2006, more than 40 maintenance cycles have been carried out. In this relatively short time, the EA repository has already provided important insights into the company's enterprise architecture that were unavailable so far. First, it has provided a holistic view which not existed before. Secondly, it has provided a means of centrally storing relevant information about enterprise architecture artifacts and their relationships so that various inconsistencies could be identified. Up to now, more than 100 inconsistencies have been identified and addressed by respective change requests. Third, and definitely most important, the EA repository has enabled a number of analyses that were either unavailable or were difficult and costly to perform before. More than 40 analyses related to 10 different application scenarios have been performed since the first release of the repository. These architectural and risk analyses helped to highlight a number of significant risks and issues relating to strategic options, redundancy and business continuity.

## 6 Conclusions and Future Work

One major finding from implementing a federated approach to maintain EA models is that the integration of existing models from specialized architectures strongly influenced the acceptance of EA as a management tool. For the EA stakeholders it became a very powerful tool since it provides valuable insights in the current and future architecture that were not available before. Due to the organizational fragmentation which most large

service companies show, particularly the different relationships between the specialized architectures were not available for analysis before. The acceptance of this solution among the providers of the specialized architectures is very high because they remain the owners of the respective architecture models.

Another insight gained from the implementation is worth mentioning: The integration of model data from specialized architectures into the EA repository is an ongoing process rather than a one-time effort. It is necessary to monitor the quality of model data from source systems continuously – particularly regarding their consistency.

From our experience, further research is needed for integrating the maintenance process into a holistic EA management and usage process. Furthermore, tool support needs to be extended. In particular, the process of loading specialized architecture model data needs more automation. However, the automation of model data updates may not be reasonable for every specialized architecture model. Especially in the case of rarely changing models there may not be a business case for an automation of updates. Criteria influencing the cost-benefit ratio of an automated approach need to be elaborated.

## References

- [Az05] Aziz, S. et al.: Enterprise Architecture: A Governance Framework - Part I: Embedding Architecture into the Organization. Infosys Technologies Ltd., 2005.
- [Az06] Aziz, S. et al.: Enterprise Architecture: A Governance Framework - Part II: Making Enterprise Architecture Work within the Organization. Infosys Technologies Ltd., 2006.
- [Ba96] Bach, V. et al.: Enabling systematic business change integrated methods and software tools for business process redesign. Vieweg, Braunschweig 1996.
- [Be05] Bernard, S. A.: An Introduction to Enterprise Architecture: Second Edition. Authorhouse, Bloomington, IN 2005.
- [BK05] Bittler, R. S.; Kreizmann, G.: Gartner Enterprise Architecture Process: Evolution 2005. Gartner Inc., Stamford, CT 2005.
- [Br03] vom Brocke, J.: Referenzmodellierung - Gestaltung und Verteilung von Konstruktionsprozessen. Logos Verlag, Berlin 2003.
- [BS02] Buchanan, R. D.; Soley, R. M.: Aligning Enterprise Architecture and IT Investments with Corporate Goals. OMG Whitepaper, Object Management Group, Needham 2002.
- [Bu06] Bucher, T. et al.: Analysis and Application Scenarios of Enterprise Architecture - An Exploratory Study. In: Proceedings, EDOC Workshop on Trends in Enterprise Architecture Research (TEAR 2006) within The Tenth IEEE International EDOC Conference (EDOC 2006), Hong Kong 2006.
- [Ci01] Chief Information Officer Council: A Practical Guide to Federal Enterprise Architecture, Version 1.0. 2001.
- [Ci05] Chief Information Officer Council: Federal Enterprise Architecture Reference Model Maintenance Process. 2005.
- [Ci99] Chief Information Officer Council: Federal Enterprise Architecture Framework, Version 1.1. 1999.
- [Da93] Davenport, T. H.: Process Innovation - Reengineering Work through Information Technology. Harvard Business School Press, Boston 1993.
- [DL04] ter Doest, H.; Lankhorst, M.: Tool Support for Enterprise Architecture - A Vision. Telematica Instituut, Enschede 2004.

- [Dv01] Department of Veterans Affairs: Enterprise Architecture: Strategy, Governance, & Implementation. 2001.
- [GKC06] Garg, A.; Kazman, R.; Chen, H.-M.: Interface descriptions for enterprise architecture. In: Science of Computer Programming 61 (2006) 1, pp. 4-15.
- [He04] Hevner, A. R. et al.: Design Science in Information Systems Research. In: MIS Quarterly 28 (2004) 1, pp. 75-105.
- [If99] IFIP-IFAC: GERAM: Generalised Enterprise Reference Architecture and Methodology, Version 1.6.3. IFIP-IFAC Task Force 1999.
- [Im97] The Information Management Group (Ed.): PROMET BPR: Methodenhandbuch für den Entwurf von Geschäftsprozessen, Version 2.0, St. Gallen 1997.
- [Jo06] Jonkers, H. et al.: Enterprise Architecture: Management tool and blueprint for the organisation. In: Information Systems Frontier (2006) 8, pp. 63-66.
- [La05] Lankhorst, M.: Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer, Berlin et al. 2005.
- [Ma03] Malone, T. W. et al.: Tools for Inventing Organizations: Toward a Handbook of Organizational Processes. In: Malone, T. W.; Crowston, K.; Herman, G. A. (Eds.): Organizing Business Knowledge - MIT Process Handbook. The MIT Press, Cambridge Massachusetts, London England 2003, pp. 13-38.
- [Ma99] Malone, T. W. et al.: Tools for Inventing Organizations: Toward a Handbook of Organizational Processes. In: Management Science 45 (1999) 3, pp. 425-443.
- [Mi79] Mintzberg, H.: The Structuring of Organizations: A Synthesis of the Research. Prentice-Hall, Englewood Cliffs, NJ 1979.
- [Mü06] Müller, S. et al.: Integratives IT-Architekturmanagement. In: Hasselbring, W.; Reussner, R. (Eds.): Handbuch der Software-Architektur. dpunkt, Heidelberg 2006, pp. 187-210.
- [Og03] The Open Group (Ed.): TOGAF (The Open Group Architecture Framework) Version 8.1 "Enterprise Edition". San Francisco, CA 2003.
- [Ös95] Österle, H.: Business Engineering: Prozess- und Systementwicklung, Volume 1: Entwurfstechniken. 2nd edition, Springer, Berlin et al. 1995.
- [PL77] Pfeffer, J.; Leblebici, H.: Information Technology and Organizational Structure. In: Pacific Sociological Review 20 (1977) 2, pp. 241-261.
- [RB06] Ross, J. W.; Beath, C. M.: Sustainable IT Outsourcing Success: Let Enterprise Architecture Be Your Guide. In: MIS Quarterly Executive 5 (2006) 4, pp. 181-192.
- [Ro94] Rood, M. A.: Enterprise Architecture: Definition, Content, and Utility. In: Proceedings, Third Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises 1994, pp. 106-111.
- [RWR06] Ross, J. W.; Weill, P.; Robertson, D. C.: Enterprise Architecture as Strategy: Creating a Foundation for Business Execution. Harvard Business School Press, Boston 2006.
- [Sc04] Schekkerman, J.: How to Survive in the Jungle of Enterprise Architecture Frameworks: Creating or Choosing an Enterprise Architecture Framework. 2nd edition, Trafford Publishing, Victoria, British Columbia 2004.
- [Sc05a] Schekkerman, J.: The Economic Benefits of Enterprise Architecture: How to Quantify and Manage the Economic Value of Enterprise Architecture. Trafford Publishing, Victoria, British Columbia 2005.
- [Sc05b] Schekkerman, J.: Trends in Enterprise Architecture 2005: How are Organizations Progressing? , Institute for Enterprise Architecture Developments, Amersfoort 2005.
- [Sc99] Scheer, A.-W.: ARIS - Business Process Frameworks. 3rd edition, Springer, Berlin 1999.
- [SE07] Smith, M. L.; Erwin, J.: Role & Responsibility Charting (RACI). [http://www.pmforum.org/library/tips/pdf\\_files/RACI\\_R\\_Web3\\_1.pdf](http://www.pmforum.org/library/tips/pdf_files/RACI_R_Web3_1.pdf), last access 29.03.2007.

- [SH93] Spewak, S. H.; Hill, S. C.: Enterprise Architecture Planning - Developing a Blueprint for Data, Applications and Technology. John Wiley & Sons, New York 1993.
- [Ve01] Veasey, P. W.: Use of enterprise architectures in managing strategic change. In: Business Process Management Journal 7 (2001) 5, pp. 420-436.
- [Ve91] Venkatraman, N.: IT-Induced Business Reconfiguration. In: Scott Morton, M. S. (Ed.): The Corporation of the 1990s. Information Technology and Organizational Transformation. Oxford University Press, New York 1991, pp. 122–158.
- [Wa05] Wagter, R. et al.: Dynamic Enterprise Architecture: How to Make It Work. John Wiley & Sons, Hoboken, New Jersey 2005.
- [WF06] Winter, R.; Fischer, R.: Essential Layers, Artifacts, and Dependencies of Enterprise Architecture. In: Proceedings, EDOC Workshop on Trends in Enterprise Architecture Research (TEAR 2006) within The Tenth IEEE International EDOC Conference (EDOC 2006), Hong Kong 2006.

# **EA Model as central part of the transformation into a more flexible and powerful organisation**

Stefan Gerber, Uwe Meyer and Claus Richert

Personal & Corporate Banking IT and Operations  
Deutsche Bank AG  
Alfred-Herrhausen Allee 16-24  
65760 Eschborn

**Abstract** This report introduces an approach how Enterprise Architecture (EA) design can be deployed in a large financial organisation for strategic transformation. Our EA design embraces all main components of the business organisations, its information systems and the way they work to achieve business objectives. In order to tackle such EA design and its deployment, governance, design and measurement principles are required to keep EA consistent and avoid misunderstandings among stakeholders. Since EA focuses on a holistic view of the organisation, full EA deployment is risky due to cost and organisational impact. Therefore we use an iterative approach within EA deployment that will be considered as an assessment process evaluating the whole IT-landscape of a certain CIO area. There are metrics used which allow the identification of transformation objects and these will be reworked in different structures by using architectural principles and then integrated into EA. Finally the existing EA will be evaluated (together with transformation object) by EA design principles and either the transformation will be rejected or design principles will be adopted. In order to make this model operative it is embedded in an architecture organizational structure which is independent from the organizational structure of the enterprise.

## 1 Introduction – EA governs IT towards better business alignment

To become a business enabler and provide faster time to market within a strong resilient banking environment, is the key focus of EA introduction [WG004]. EA is a discipline which synchronizes the business strategy with the IT strategy. Hence, EA can not be a one-time effort, but is subject to the same change as the enterprise itself. Mergers/Acquisitions, growth strategies or consolidation efforts will heavily impact the way EA is conducted in an enterprise. EA should be a major driver in adapting the IT landscape, which mostly consists of applications and infrastructure, supporting the business processes. Unfortunately, the lifetime of applications in most cases is much longer than the average time between business and IT strategies change. Thus, a flexible approach to EA is needed to drive these changes towards the implementation of the strategies.

Our approach is to establish a hierarchy of business-aligned Enterprise Architects and functional and non-functional domains. The functional domains (e.g. Cash Management, Loans Management) cover the IT landscape from a business point of view, whereas the non-functional domains deal with overlapping concerns such as security or integration. On the next level, projects build new business solutions. This approach yields a strong business alignment through business involvement.

In addition, the division of the architecture into domains helps to reduce the complexity and assigns responsibilities based on knowledge. It is not an option to deploy EA in one big bang due to the complexity, therefore an iterative approach is required – and this will be described in this report.

## 2 EA design delivery structure and development approach

### 2.1 Delivery structure of the target architecture and design principle

Architectural design structures as they are suggested by the Zachman Framework [Z1987] or The OPEN ARCHITECTURE GROUP (TOGAF) [TO002] follows a fix structure of multiple layers. As most studies from industrial practise show, adaptations on these models are made to bring case related design into generic design. A full implementation of such a model will often be rejected because of time and costs.

Our design uses a simple 4-layered architecture that has been suitable for architectural design in the literature [BFK06], [KAV05], [WG004], [JH007].

- **Business architecture:** Value networks, relationships to customer and supplier, target market segments, offered services, organizational & strategic business goals and strategic projects
- **Process architecture:** Business processes, organizational units, responsibilities, performance indicators and information flows



- **Integration architecture:** Enterprise services, application clusters, integration systems and data flows
- **Software architecture:** Fundamental software organization artefacts, software services and data structure

Practical experiences shows, architectures structured in 4 layers spoil detail and are too granular or generic [MP006],[BSV07]. This causes problems in communication and planning due to various audiences involved. Therefore to avoid misunderstanding between the different viewpoints, architectural layers will be considered with different levels of abstraction.

- **Enterprise level:** The enterprise level is the highest abstraction level where all strategic decisions regarding business, operations and IT will be described for a particular closed section of the enterprise. Instead of seeing the whole entity as an enterprise we believe that major business lines as e.g. Private and Corporate Business, Wealth Management or Investment Banking are the appropriate level of abstraction. Therefore the number of enterprise areas should be limited to 3 to 5 in maximum.
- **Domain level:** On domain level the strategic goals from the enterprise will be detailed and deployed on domain specific operating models, processes, applications and/ or infrastructure. On this level all guidelines and principles will be defined. Therefore between enterprise and domain level there is a 1:n relationship to achieve a higher granularity in architecture design. For example, the business line Private and Corporate Business will consist of domains as Financing, Investments, Payments, and Current Accounts etc. At minimum 4 domains up to a maximum of 8 domains should belong to one enterprise area.
- **Solution level:** The scope of the solution level embraces all applications and their related technical systems. Projects will be performed and delivered out of this level. At this level the detailed technical and business design, software artefacts behave as user interfaces, storage components, computation functions, connectivity components, security components or process components are developed, maintained, tested etc

Our architecture design seems feasible for enterprise architecture design. We are able to identify how our architecture may/will be affected by changes or new requirements from either business or technology. However, we need the alignment of the different model dimensions with respect to specific techniques or methods that we have to keep unique in our architectural design; our model may have some drawbacks because of its strict hierarchical structure (i.e. Enterprise services on each layer will be implemented differently).

Therefore our architectural design structure will be extended by an additional dimension, so called interdisciplinary dimension. Through this dimension we provide design principles which have an impact on all different architecture levels according to specific scopes. This dimension is required to cover technological aspects; therefore we have defined following scopes:

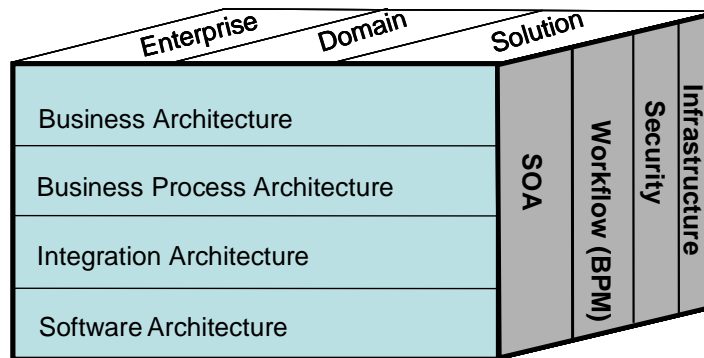


Figure 1: Enterprise Architecture Delivery Structure

**SOA:** All aspects of Service-Oriented Architecture from service discovery to deployment including methodology, training, coaching, governance and technology.

**Workflow:** Methodology/framework (including governance) and platform definitions to achieve consolidated automation and monitoring of document-centric workflows to become the workflow competency partner to business.

**Security:** Implementation of application security in a cost-efficient, consistent and interoperable manner meeting requirements out of IS Policy

**Infrastructure:** Provides a framework on how to architect applications and services to make best use of infrastructure. This covers infrastructure on various tiers such as Operating System, Persistence and Application Services.

The complete delivery structure of our enterprise architectural design model finally considers architecture artefacts in three dimensions: abstraction level (3), architectural layers (4) and interdisciplinary layers (3-n) by following this structure, our EA design implements various relationships between architectural objects e.g. processes or application such that multidimensional behaviour analysis can be performed. Functional or process relationships towards applied technology or applications can be obtained and reasoned with the help of such analysis.

## 2.2 EA design development approach using assessments

Architecture thinking is now well established in many organisations. However, efforts estimates and costs of full EA design are often underestimated and consequently prevent firms to succeed in their architectural objectives [RSV07], [KAV05], and [ABB07]. It is important to achieve objectives that were put for the architectural program in the beginning despite the constraints of budget etc. Many firms try to run independent projects with different scope and topics in order keep efforts feasible. However, due to differences in scope and level some overheads are necessary to align different projects. Although these projects use a predefined structure for the delivered architectural design, it may not be possible to get comparable results, which are interpreted the right way and easily integrated into the general EA model without additional effort. In the literature [RSV07] some investigations are made on how to justify maturity and alignment capability of a given architectural design. Based on such assessment it may become possible to integrate architectural designs at different levels into a single enterprise architecture model at feasible costs.

Contrary to previous approaches, in our approach EA design will be deployed in so-called transformation objects with the need to transform the architectural structure in order to improve business enablement or IT quality. Such transformation objects may be identified on different architecture layers or abstraction levels by using an assessment model. The assessment model aims to evaluate the strategic potential of the transformation objects for both business and IT before spending any effort on EA design. Thus architecture design processes for different transformation objects will always be aligned firstly with each other and secondly with strategic and architectural principles striving for the architecture program. As assessment is the central part of our approach, the full process contains some further steps needed to prepare the assessment and finally implementing the transformation objects as well as keeping them in accordance with EA principles. The entire EA development process will be implemented by a V- model as shown in the figure 2:

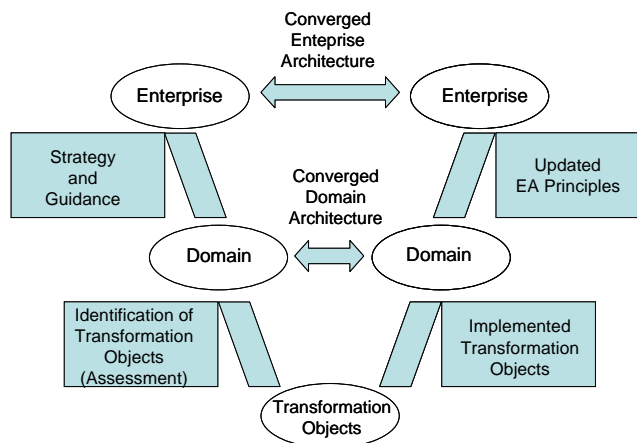


Figure 2: V-Model of Enterprise Architecture Development

The assessment uses multi-dimensional evaluation approach, based on this approach the IT landscape will be measured according to their Business contribution, Technical Quality and Costs. Each of the measurement dimensions (e.g. Business contribution) have a certain metrics applied to identify opportunities, bundle them into transformation objects for improvements, quantify their rank and finally define the high level master plan for architectural integration. Assessment of market packages or components-off-the-shelf sometimes is feasible too, as very often adoption and integration into the existing environment needs architecture as well.

An assessment can be viewed as snapshot at a certain time. Repetition of the assessment with the same candidate at a later stage is foreseen in this model to prove the impact of architectural changes. It has to be considered that due to changes in markets, technical or organisational realignments etc., not only change transformation objects but will determine the future results.

Major advantage of this approach is the involvement of the business units in EA design and the alignment of the business strategies with the IT strategy.

### 3 Organizational structure making EA work

With EA design we identify the areas where we need to transform the architecture of our solutions and better manage the governance of our portfolio. Therefore to achieve this and to better support businesses in their growth targets introduction of EA organisation is a major step towards transformation into a more flexible and service oriented organization. One of the core pillars of this structure will be dedicated Enterprise Architects for each business line. Enterprise architects will work closely with business partners to set the strategic direction of the overall architectural business landscape on the basis of the inputs from the underlying domain. The basic architectural work will be done within domains. Therefore for each domain a domain architect is in charge managing the work of several solution architects. The entire organizational structure of EA with all responsibilities and deliverables is defined in the following table:

Level	Task	Deliverables
<b>Enterprise Architect</b>	Business Strategy	Business Vision/Strategy consisting of goals and objectives for Client and portfolio of investments, roadmap of initiatives for the next 3-5 years, Competitive analysis reports, Business capability roadmaps and initiatives for next 3-5 years
	IT Strategy	IT strategy consisting of goals and objectives for the IT organization over the next 3 to 5 years, given current organization performance and business support needs, technology initiatives proposal (EA input to business strategy), updated IT strategy/vision

<b>Level</b>	<b>Task</b>	<b>Deliverables</b>
	Governance, Portfolio Definition, Organization & Management	Architecture governance model, rules of engagement, roles and responsibilities, escalation process, business portfolio model, portfolio definitions (scope of portfolio in functional or business process terms), assets assigned to each portfolio and classified (e.g. core, strategic, mission critical, legacy), portfolio strategy, roadmap of programs & projects.
	Architecture Metrics and Performance Management	Architecture marketing and education materials, EA communications plan, balanced scorecard framework, architecture measurement system, balanced scorecard report, performance action plan
<b>Domain Architect</b>	Domain Architecture Planning	IT Strategy consisting of goals and objectives for technology over the next 3 to 5 years given technology and industry trends, Current state architecture, relevant business & technology imperatives, guiding principles develop/refresh, gap analysis, future state architecture vision, implementation roadmaps
	Principles and Requirements	Guiding principles for IT, EA, design, deployment, policies, data, security, etc.
	Domain Governance and Demand Management	Process for reviewing solution architectures for compliance with standards, roadmaps and goals for reusability; includes approval criteria, list of required/optional artifacts at each phase/gate
	Business Architecture Development	Business value chain model, business capabilities model, business process maps, business information model
	Vendor Relationship Management	List of strategic IT vendors, Vendor evaluation criteria and metrics, List of vendor relationship managers; Manage Non-strategic vendors
<b>Solution Architect</b>	Solution Architecture Design	Solution requirements, solutions architecture plan with potential projects, solution architecture plan considering portfolio architecture roadmap, technical architecture roadmaps and standards, business case or value case, solution architecture design document
	Architecture Requirements and Design	Technical reference model, engineered patterns (i.e. web-user-interface design, single sign on, rules engine topology, static web content delivery, web personalization, etc) document management patterns, design patters, naming conventions, code frameworks, etc., populated patterns from actual projects/solutions

This organizational structure will be embedded in a governance framework to guide the work and ensure the quality of deliverables of all involved parties. The following characteristics are positioned here to highlight both the value and necessity for governance:

**Discipline:** All involved parties will have a commitment to adhere to procedures, processes and authority structures established by the organization

**Transparency:** All actions implemented and their decision support will be available for inspection by authorized organization and provider parties

**Independence:** All processes, decision-making, and mechanisms used will be established to minimize or avoid potential conflicts of interest

**Accountability:** Identify groups within the organization, e.g. Governance Boards, who take actions or make decisions, are authorized and accountable for their actions

**Responsibility:** Each contracted party is required to act responsibly to the organization and its stakeholders

#### **4 EA's evolution over time declares the roadmap for IT convergence**

The five architecture views described in the previous chapter are usually all affected during the change of a transformation object. This way, the change of the transformation object contributes to IT Convergence on multiple levels.

**1. Infrastructure Architecture:** Deutsche Bank's Technology Roadmap classifies all infrastructure components into different lifecycle states (Invest, Maintain, Disinvest, and Unsupported) according to the strategic fit and the maturity respective state of support offered by the vendor and the internal Engineering and Operations. Close monitoring of the implemented technology allows stringent management of the infrastructural components and minimizes the risk of malfunction due to the use of unsupported technology. Most importantly, infrastructural standardization is the key element towards reduction of heterogeneity and leading to simplification which in-turn reduces cost.

**2. Software Architecture:** Software artefacts are governed through above mentioned standardization and by the respective Domain Architects who works with the solution project teams to achieve convergence to the transformation objectives. This model ensures that architectural compliance is not an ex-post event; rather it is the result of a pro-active engagement.

**3. Integration Architecture:** How to integrate applications with each other is governed by a Domain Architect for Service-oriented Architecture and Integration. Guidance is provided through related documentation and reference architectures. Through standardization in the Integration Architecture interoperability will be increased and future integration becomes easier.

**4. Process Architecture:** Process Architecture is as well a major focus area: A Domain Architect for Business Process Management provides the guidance and governance around Business Process Management and Modelling. Standard tools and methodologies have been defined. This approach drives, together with the process owners in business, the modelling (and automation) practice, standardisation and convergence of the process landscape itself. This is an important step towards a service-oriented enterprise.

**5. Business Architecture:** Alignment between business and IT is achieved through collaboration in Business Enterprise Architecture Forums and as well on the next level (Domain Forums). This dialogue between decision-makers in the Business ensures the convergence on a strategic level and transformation objects and their implementation can be discussed here.

In case, the transformation candidate has been rejected, EA principles should be reviewed and modified if required. This ensures that EA principles can be adapted to strategic or environmental changes. This feedback cycle is a critical element in EA evolution driving the IT convergence.

## **5 Conclusion**

The approach shown for enterprise architecture in a large financial organisation consists of elements that all need to fit together to realize the envisioned strategic transformation towards a service-oriented enterprise.

The division of IT into domains is the pre-requisite for a divide-and-conquer strategy that allows for effective architecture governance. We have explained how governance, identification and analysis of transformation candidates are performed and jointly contribute to the application and evolution of EA.

Overall, our approach is a suitable way to iteratively evolve Enterprise Architecture and the IT landscape towards with more convergence to achieve a service oriented enterprise.

## References

- [ABB07] F. Arbab, F. de Boer, M. Bonsangue, M. Lankhorst, E. Proper, L. Van der Torre: Integrating Architectural Models Symbolic, Semantic and Subjective Models in Enterprise Architecture, Enterprise Modelling and Information Systems Architectures, Volume 2 No.1, May 2007.
- [BFK06] T. Bucher, R. Fischer, S. Kurpjuweit, R. Winter: Analysis and Application Scenarios of Enterprise Architecture: An Exploratory Study: Proceedings of the 10<sup>th</sup> IEE International Distributed Object Computing Conference Workshop (EDOCW), 2006.
- [JH007] Marijn Janssen, Kristian Hjort-Madsen: Analyzing Enterprise Architecture in National Governments: The cases of Denmark and the Netherlands, Proceedings of the 39<sup>th</sup> Hawaii International Conference on Systems Sciences, Jan. 2007.
- [KAV05] Stephen H. Kaisler, Frank Amour, Michael Valivullah: Enterprise Architecting: Critical Problems, Proceedings of the 39<sup>th</sup> Hawaii International Conference on Systems Sciences, 2005.
- [MP006] Mirja Pulikkinen: Systematic Management of Architectural Decisions in Enterprise Architecture Planning, four Dimensions and three abstraction Levels, Proceedings of the 39<sup>th</sup> Hawaii International Conference on Systems Sciences, 2006.
- [RSV07] Bas van der Raadt, Raymond Slot, Hans van Vliet: Experience Report: Assessing a Global Financial Services Company on its Enterprise Architecture Effectiveness using NAOMI, Proceedings of the 39<sup>th</sup> Hawaii International Conference on Systems Sciences, Jan. 2007.
- [TO002] The Open Group: The Open Group Architecture Framework (TOGAF) Version 7 "Technical Edition", Version 8 "Enterprise Edition". Document Nr. 1911 December 2002. <http://www.opengroup.org/togaf/>
- [WG004] Wolfgang Gaertner, Ansatz für erfolgreiche Enterprise Architecture im Bereich Global Banking Division/Global Transaction Banking IT and Operations der Deutschen Bank, Wirtschaftsinformatik, 4/2004.
- [Z1987] Zachman, J.A, A framework for information systems architecture. IBM Systems Journal, Vol. 26, No 3, 1987, IBM Corporation, pp. 276-292



# Generating Visualizations of Enterprise Architectures using Model Transformations

Sabine Buckl, Alexander M. Ernst, Josef Lankes,  
Christian M. Schweda, André Wittenburg

{buckls, ernst, lankes, schweda, wittenbu}@in.tum.de

**Abstract:** Giving account to the importance of enterprise architecture (EA) modeling, this article sketches common issues in visualization handling that we came across during an extensive survey of the existing tool support for EA management in 2005. We introduce the research project software cartography, in which we develop an approach for EA modeling including a method for the automatic creation of EA models and visualizations. This approach is based on model transformations, which we use to link the data to be visualized and their graphical representation, thereby circumventing the error prone and time consuming task of manual creation of the visual models. A brief overview of a prototypic implementation of this approach complements the theoretic findings and illustrates applicability for visual modeling and documenting the EA.

## 1 Motivation

With the growing importance of enterprise architecture (EA) management currently experienced in research [LW04] and in practice [Jam05], methods for documenting, evaluating, and planning the application landscape as part of the EA gain increasing attention. This is reflected by various approaches, which try to establish and standardize languages for modeling the EA, furthermore complemented by a number of vendors claiming the emerging market of EA management tools. Nevertheless, many of these tools show common weaknesses, especially regarding the approach used for creating visualizations of the EA or the application landscape, as we found out during an extensive survey [seb05] conducted by sebis. Such visualizations, used for documenting, evaluating, and planning the application landscape make up the focus of the research project *Software Cartography*, which this paper originates from.

In this project, we discovered a large number of different visualizations for application landscapes, which we refer to as *software maps*. An exemplary software map used at one of our project partners is given in Figure 1. The figure is made illegible due to the fact that it contains confidential information. Nevertheless, the figure shows the inherent complexity an approach for generating visualizations of enterprise architectures has to cope with. The software map originates from an insurance company and visualizes about 160 application systems hosted at the headquarter, which are used worldwide. The original map is commonly used as printout in DIN A0, within presentations, and is available at the corporate intranet.

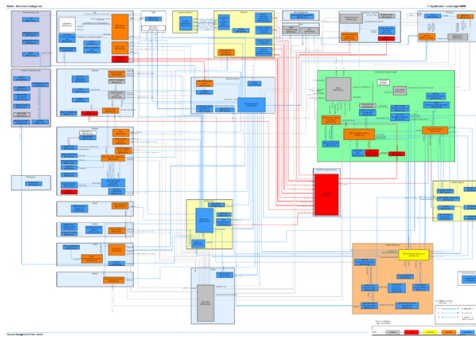


Figure 1: Exemplary software map of an insurance company

In order to discuss the requirements an approach for the generation of visualizations of EAs must satisfy, an anonymized software map similar to the one of the insurance company is shown in Figure 2. This visualization shows organizational units of a fictitious department store as rectangles, nesting the business applications hosted at the specific organizational unit represented by smaller rectangles. No established method for the creation and maintenance of such visualizations yet exists. Furthermore, most of the EA management tools show only basic capabilities in the context of automated positioning [seb05]. Within the development of such a method the following issues have to be considered:

- The manual creation of the visualizations of the EA is an error prone and time consuming task, that leads to software maps containing aged data. Caused by the missing link between the present data and the visualization, no automated creation process for the visualization is available to ensure the timeliness of the visualized data.
- The EA management tools commonly provide the user with the possibility to introduce visual elements without defined semantics in the context of the visualization, thereby effectively disconnecting the visualization from the respective data.

We subsequently detail on the topic of EA modeling, presenting an approach, complemented by a prototypic tool implementation, which we regard to be suitable for addressing these issues. Thereby, the approach is based on a technology originating from the field of model driven development (MDD): *model transformation*. This article especially focuses on the method for creating visualizations of the EA by model transformation and provides information, how a tool could actually implement this method. Thereby, the error prone and labor intensive task of manual creation of these visualizations is eliminated.

The remainder of the article is structured as follows. As a starting point, software cartography as an way to support EA modeling with visual models is presented in Section 2 as well as an approach using model transformation to create the necessary visual models. The following Section 3 shows the application of our approach by providing information on a prototypic tool implementation. Section 4 emphasizes on different approaches taken in the context of EA modeling as well as on aspects of visualization consistency. Finally, sec-

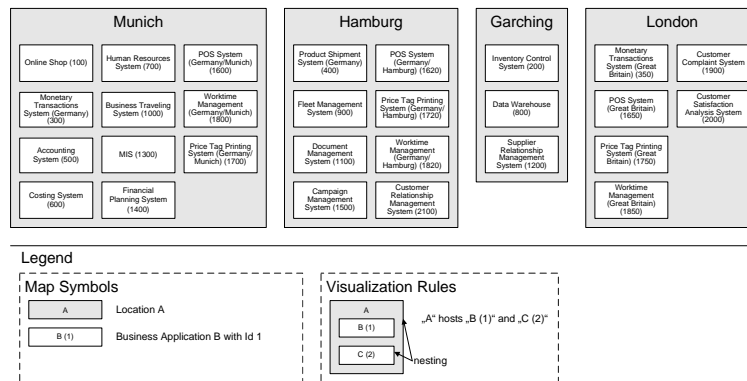


Figure 2: Exemplary software map

tion 5 provides some conclusions resulting from the taken approach and sketches aspects of further research in this field.

## 2 A model transformation approach

Our approach to EA modeling uses concepts and notions originating from the field of cartography. Maps in the context of cartography can be categorized into two different map types: *topographic maps* and *thematic maps* [KO96]. Topographic maps mainly deal with geographic information, whereas thematic maps show spatial information on a topographic map, as e.g. the results of a political election. In the context of EA modeling, visualizations resembling the buildup of thematic maps can be considered to be important, as they can be used to visualize different aspects of the enterprise. These visualizations, called *software maps*, are subject of research in our project software cartography. Aspects in the context of EA modeling that can be used to support the documentation, planning, and evaluating of the application landscape can be found in [MW04]. Thereby, metrics that point out aspects can be visualized on software maps to address specific concerns. In our research project, we gathered different visualizations of the EA and categorized them into three different types [Wit07]:

- A *cluster map* is a type of software map that uses positioning to show how objects (e.g. applications) are grouped into larger logical units (e.g. organizational units). Thereby, the graphical representation of the object is clustered into the the representation of the logical unit. An example for a software map of type cluster map is shown in Figure 2.
- A *cartesian map* is characterized by elements that are aligned along an x- and an y-axis. Two prominent examples of a cartesian map exist. Firstly, the process support map, which utilizes positioning to show which business processes (y-axis) are

supported by which application and used at which location (x-axis). Secondly, the time interval map, which is closely related to Gantt-like diagrams, as it uses bars for representing the life cycle on the x-axis (representing periods of time) of objects (e.g. applications) on the y-axis.

- A *graph layout map* is a map using a typical nodes-and-edges buildup, not exerting additional restrictions on positioning to convey information. Therefore, the positioning is for example used for minimizing the numbers of lines crossing.

To support the visualization of different aspects, as e.g. technical aspects or economical aspects on a software map [LMW05], the *layering principle* as shown in Figure 3 can be utilized.

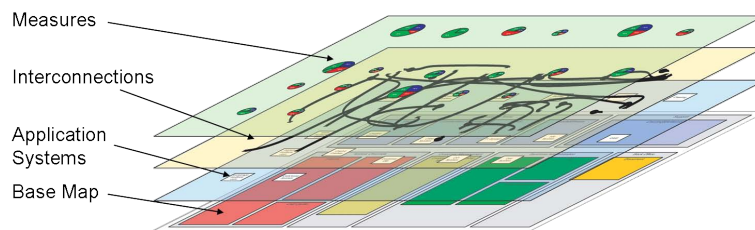


Figure 3: Layered architecture of a software map

The exemplary software map consists of a base map including organizational units, and multiple layers, which are used to visualize relationships between different objects. In Figure 3, the layers contain applications on the first layer, interconnections representing a technical aspect on the second layer as well as measures on the third layer, visualizing operational or economical aspects. Thereby, each layer has a reference layer to which the elements correspond.

As described above, we pursue an approach for EA modeling based on model transformation in order to ensure the consistency between models (e.g. data in an EA management repository) and visualizations of the EA. Therefore, a strict separation of the content to be visualized - the *semantic model* - and its representation - the *symbolic model* - is required. Additionally, a well-defined link between these models - the *transformation* - is needed. Figure 4 shows the basic idea of the model transformation approach. Subsequently, the individual concepts are explained in detail.

## 2.1 Semantic model and information model - the left side

The semantic model and the information model deal with the information describing the EA and its structure, thereby, the different models represent different levels of abstraction, similar to the notion of MOF (e.g. *class* and *instance*). The focal point of the semantic model lies on the actual *information objects*, which describe the application landscape

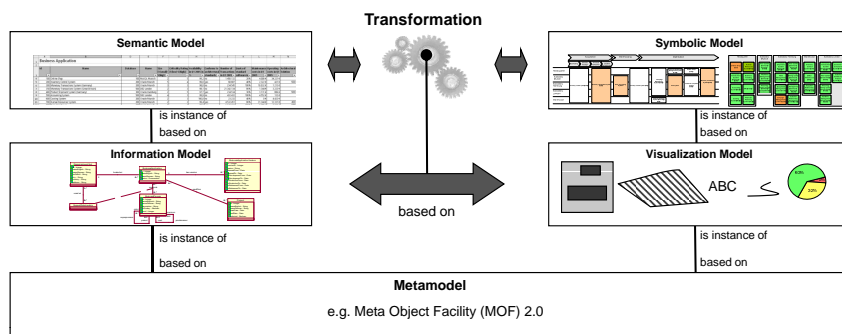


Figure 4: Basic principle of the software cartography method

irrespective of its representation. These information objects are instances - in terms of object orientation - of the classes of the information model, thus the information model is the metamodel on which the semantic model is based.

To exemplify the two tiered structure of the *left* side, we refer to the cluster map introduced in Section 1, i.e. the respective information about the EA contained therein. This information can be summarized as "which location hosts which business application". "Munich", for example, which is an instance of `Location`, hosts among others "Online Shop (100)", an instance of `BusinessApplication`. Figure 5 shows some of the information objects, which are instances of the classes from the information model in Figure 6.

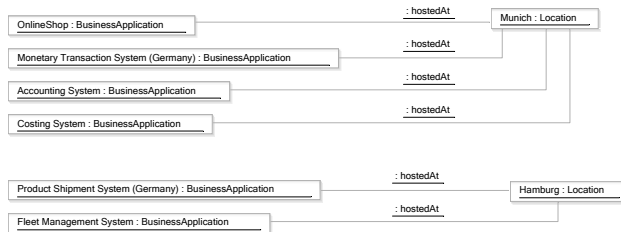


Figure 5: The semantic model containing some information objects presented in the cluster map



Figure 6: The corresponding information model

The respective information model thus contains the classes `BusinessApplication` and `Location`, related by the association `hostedAt`. The attributes of the classes in the information model are not described in detail here, as only three of them are shown exemplarily. A more detailed description of information models and their related visualizations for EA management can be found in [BEL<sup>+</sup>07].

## 2.2 Symbolic model and visualization model - the *right side*

In order to provide means for describing visualizations, as the cluster map shown in Figure 2, we introduce a visualization model containing elements representing graphical concepts. These graphical concepts may on the one hand be *map symbols*, as e.g. the rectangle and on the other hand be *visualization rules*. These rules exert certain demands on the positioning, size, or overall appearance of the map symbol instances, as e.g. the `Nesting` rule, used in the exemplary visualization, demands that a symbol representing a business application is fully contained in the outer symbol. Utilizing these concepts, the visualization can be described by a symbolic model (see Figure 7), that consists of instances from the exemplary visualization model (see Figure 8). Nevertheless, it must be noted, that there exist more visualization rules, even in this simple example. An example is the rule demanding the different symbols representing business applications not to intersect each other. A complete model, able to describe visualizations as introduced above, is contained in [ELSW06].

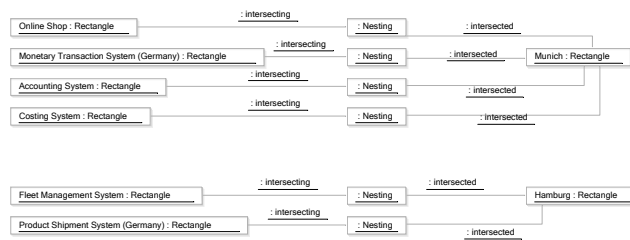


Figure 7: The symbolic model containing some visualization objects of cluster map



Figure 8: The corresponding visualization model

The object-oriented visualization model, alluded to above, greatly leverages the model transformation approach, but nevertheless is not capable of giving a strict definition for the visualization specific semantics of the map symbols and visualization rules. Therefore, we complement each class of the model with an expression in predicate calculus, describing the graphical implications in an unambiguous way. These definitions, further detailed in [ELSW06], can be used for computing the actual visualization from a symbolic model. Such a system might pursue different approaches for the computation. An exemplary one is outlined in section 3.

### 2.3 Model transformation and metamodel - the *middle*

To allow an automated creation of visual models of the application landscape and to ensure the consistency between these models and the underlying data, a link between the *left* side, representing the information and the *right* side, the representation, is required. This link is created by a transformation, which translates the information objects of the semantic model into visualization objects of the symbolic model. Selecting a transformation language specification, the concepts used in information models for EA management and the bidirectionality of the transformation, to allow changes in the semantic model by interacting with the visualization, should be considered. Figure 9 gives a short example of a transformation, resembling a notation as proposed by MOF *Query, View, Transformation (QVT)* [OMG05a].

```
rule OrgUnit2Rectangle {
  from
    infoObject : Semantic.OrganizationalUnit
  to
    symbol : Symbolic.Rectangle (
      text = infoObject.name,
      backgroundColor = #CCCCCC
    )
}
rule BusinessApp2Rectangle {
  from
    infoObject : Semantic.BusinessApplication
  to
    symbol : Symbolic.Rectangle (
      text = infoObject.name + "(" + infoObject.id + ")"
    ),
    rule : Symbolic.Nesting (
      inner = symbol,
      outer = transforming (infoObject.hostedAt)
    )
}
```

Figure 9: Exemplary transformation rule set

Due to the fact that a common metamodel for the information model and the visualization model greatly simplifies the transformation specification, such a model is subsequently introduced. We extensively analyzed different EA management information models developed by industry partners in [Buc05], which pointed to the OMG's *Meta Object Facility*

(MOF) [OMG06] as a suitable metamodel. The MOF model contains two core packages, *Essential MOF (EMOF)* and *Complete MOF (CMOF)*, the former providing the core capabilities usually associated with object orientation, the latter extending them with advanced constructs, as e.g. constraints. However, EA management information models at our industry partners did not turn out to heavily rely on CMOF concepts, but more showed that these advanced concepts were used inconsistently. A common sense of usage only exists concerning the core concepts as contained in EMOF.

Based on the results of the analysis alluded to above, we regard EMOF to be sufficient for information modeling in the field of EA, as well as a good choice in terms of an easy mapping of models to implementation. Verifying this choice, the following section details aspects of our prototypic tool realizing the approach outlined above.

### 3 SoCaTool: a tool for enterprise architecture modeling

Subsequently, we show the applicability of the model transformation approach for generating visual models of the enterprise architecture. Therefore, we provide information on a prototypic tool, which has been developed by sebis - giving an implementation of the approach. Prior to describing the core components of the tool and their interaction in generating visualizations, we provide a summary of our basic assumptions, which greatly influenced the software architecture of the tool.

With an approach strongly centered around the usage of object-oriented models and representations thereof, a main factor is the metamodel, all these models are based on. Considerations as in Section 2.2 advocate the usage of EMOF as a common metamodel for the information model and the visualization model. An implementation of the metamodel has therefore to be incorporated in the tool. With different implementations at hand, we decided to rely on the implementation provided in the *Eclipse Modeling Framework (EMF)* [MDG<sup>+</sup>04]. This framework was chosen, as its metamodel, the *ECore*-metamodel, can be considered to be very similar to the EMOF-metamodel<sup>1</sup>. Additionally, the EMF provides serialization and editing related functionalities at "no cost", as well as an active user and developer community. From this community various extensions to the core EMF have arisen, as e.g. a support for OCL queries. With this initial choice made, the *Eclipse Rich Client Platform* further deemed to be suitable for implementing our approach, especially with the *Graphical Editor Framework (GEF)* [MDG<sup>+</sup>04] providing an easy to use system for managing and interacting with visualizations.

Based on the eclipse rich client platform, a component architecture containing four core components has been realized - complementing the approach outlined in section 2 with an implementation. Subsequently, these components are detailed.

#### Repository

The repository component is used for storing and managing object-oriented models, as e.g. the semantic model. This component also maintains the relation between a model

---

<sup>1</sup>Only minor differences concerning naming and the usage of references exist.



and its corresponding metamodel, as e.g. the information model. Concerning the set of functionalities offered by a repository, different types of repositories can be considered. Whereas the simplest type only enables reading access to the models as well as creating a completely new model from a set of objects, a more sophisticated repository would e.g. support editing operations on the objects contained. The support for multiple users acting on object-oriented models raises additional demands on a repository, especially concerning transaction related issues as well as issues concerning notification about model changes. More detailed considerations on the functionalities supported by a repository can be found in [OMG04].

As the prototypic implementation neither needs transaction support nor notification capabilities, a simple file-based repository has been chosen, thereby, every object-oriented model is serialized as a single xml-file. Nevertheless, this repository is used via the *eclipse emf Resource*-interface, which is also supported by repository projects providing more functionalities, as e.g. the elver persistency project [Gro07].

### **Transformer**

The transformer component is capable of interpreting visualization definitions as rules describing the transformation from an object-oriented model to another. When analyzing the transformation rules between the semantic and the symbolic model, as outlined in section 2.3, we identified basic functional requirements, as e.g. a support for queries on the semantic model data as well as a support for parametrizing rules. Additionally, a framework for bidirectional transformations would greatly leverage the approach from section 2, as it would provide means for editing semantic model data via changes to the symbolic model. These requirements mainly focus on the expressiveness of the transformation language. Nevertheless, further requirements regarding the usage context have to be considered. This is especially important, as the transformation rules should be easily definable for users without "full-scale" programming knowledge, allowing users, as far as possible, to define auto generated custom visualizations. We deem it best, to have a graphical notation for defining these rules.

Taking into consideration languages for defining model-to-model (M2M) transformations, especially prominent in the field of MDA, the *Atlas Transformation Language* (ATL), as described in [gaLI06], is at first sight an interesting candidate. Pursuing a strongly declarative approach in notating the rules, and not providing a graphical notation for defining the transformation, some of the functional requirements stated above are met by ATL. Nevertheless, ATL has only a limited support for querying concepts and, as with version 0.7, did not provide support for parametrized rules<sup>2</sup>.

The *Bidirectional Object Transformation Language* (BOTL) [BM03], pursuing a strongly declarative approach, provides an UML-based graphical notation for defining transformation rules. Furthermore, it leverages bidirectionality regarding the rules, as far as the operations performed during transformation do support this. Nevertheless, BOTL uses an independent metamodel, faintly "inspired" by the EMOF metamodel, leaving out concepts that are of importance in information modeling, as e.g. inheritance. Furthermore, querying and external parametrization are not directly supported.

---

<sup>2</sup>The current version of ATL does support external parametrization.

Having thus ruled out two promising transformation languages from the field of MDA, we decided to use *ECore reflection* and java code to realize a first prototypic implementation of the transformer based on "hard coded" transformation implementations. While this approach comprises obvious drawbacks concerning the simplicity of visualization definition by the user, it greatly leverages the definition of closely related visualization variants by inheritance and the utilization of object-oriented design patterns. Additionally, the maximum expressiveness of java helped us to gain further insights, which language concepts are necessary in constructing model transformation rules for defining EA management visualizations.

### **Layouter**

The layouter component, providing the capability to actually layout visualizations described as symbolic models, can be considered the core component of the prototypic tool. This component leverages the utilization of object-oriented visualization specifications and thus enables the realization of visual modeling facilities without burdening the model creator with the implementation of layouting algorithms. When relying on the concepts provided by the visualization model as outlined in section 2, the layouter is capable of calculating the positions, dimensions, and other visual parameters of symbol instances in accordance to the visualization rule instances in the symbolic model. In performing this calculation many different approaches can be pursued. Two of them have been explored in-depth in the prototypic tool implementation, which are subsequently detailed.

The first approach relies on the fact, that for every symbolic model a representation as an optimization problem can be found. This optimization problem uses the positions, dimensions, and other visual parameters of the symbol instances as variables, while constraints and target functions are derived from the visualization rule instances [ELSW06]. Solving the corresponding optimization problem is therefore equivalent to finding a valid layout for the visualization. Nevertheless, as these optimization problems are often high-dimensional as well as non-convex, specialized algorithms for solving do not commonly exist. For this reason, the first approach employed a genetic algorithm for searching an optimal solution. Due to the high genericity of such an algorithm, this approach is of limited performance.

The second approach takes advantage of the fact, that there exist recurring elements in the object-oriented symbolic models, called *patterns*. One of these patterns could e.g. be a *clustering* pattern, in which a variable number of symbol instances is demanded to be *nested* into a surrounding symbol instance, with the nested instances demanded to be *separated* from each other. This pattern is prominently used in the visualization in Figure 2. For such patterns specialized layouting algorithms can be found, which incorporate the specifics of the pattern to provide superior layouting performance. A layouter pursuing this approach has been implemented as component in the tool (see [Lau07]), performing significantly better as the genetic algorithm. Nevertheless, the layouter is limited concerning the variety of symbolic models, which can be addressed, although the most prominent types of visualizations as outlined in section 2 can be layouted.

### **Renderer**

The renderer component is used to present a layouted symbolic model in a specific output format. Concerning the format especially the PDF and the *scalable vector graphics* (SVG) format are of interest due to the inherent or potential support for layering and their vector

graphic nature. Supplementary, a renderer for direct screen output in the tool can be implemented, with additional functionalities of interest, as the option to support interactions with the rendered visualizations, e.g. via moving symbols.

In the prototypic implementation a renderer for static visualizations on screen has been implemented using the eclipse *Graphical Editor Framework* (GEF). The output of this renderer in the graphical user interfaces of the tool is shown in Figure 10, displaying an exemplary software map of type *cartesian map* as outlined in section 2.

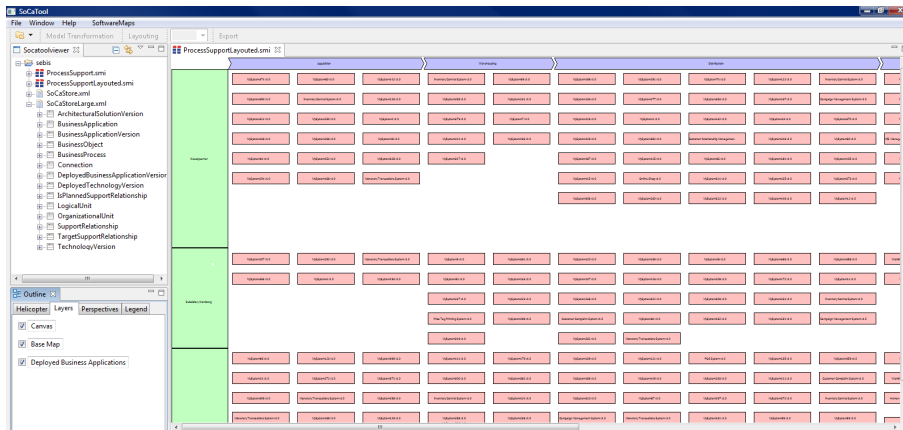


Figure 10: The GUI of the prototypic tool implementation

## 4 Related Work

With an approach for visual modeling presented above, the following section links to related work from the area of software engineering and EA modeling as well as issues regarding consistency of visual models.

In the field of software engineering, the *unified modeling language* (UML) [OMG05c, OMG05b] provides the common sense for modeling single software systems, which is lacking in the field of enterprise architecture modeling. Therefore, the attempt of *transferring* the concepts and notations of UML to EA modeling could be undertaken. Nevertheless, the specific concerns of this area of modeling are not well supported by UML, as e.g. concepts like business applications or business processes are not known. While these concepts could be introduced via UML profiles, specific diagramming semantics are not easily realizable using the concepts of UML, effectively ruling out the unified modeling language as a language for EA modeling. This fact is also reflected by the variety of different approaches for enterprise architecture modeling regarding languages, methods, and tools, which can be found in the academic community.

One approach is outlined in [vdTLD<sup>+</sup>04] and specially focuses on a formal way of defin-

ing visualizations of the application landscape. This approach relies on the concept of *signatures* to establish a well-defined relation between the visualization and the underlying model of the enterprise architecture. While this approach also considers aspects of interest in the context of visualizations, e.g. relative positioning, no simple to use notation for a model describing the visualizations is provided. Further the approach does not provide an executable way for creating visualizations from the information.

Regarding the absence of a state of the art, [Fra02] suggests another approach to enterprise architecture modeling, emphasizing the necessity to support different views on the enterprise. These views use different *special purpose modeling languages* to meet the concerns of the different stakeholders. These languages are defined in metamodels, which correspond to a common meta-metamodel to support integration. Nevertheless, as the approach is more focused on the provision of an integrated meta-metamodel for the different languages, it does not provide a method for generating the required views of the EA. The approach presented in section 2 can be seen as supportive in this context, for realizing tool support for the special purpose modeling languages and their visual models, as outlined above.

An approach centered around an EA metamodel (*information model* in our terms) can be found in [BW05]. The models contains over 50 classes and thus spans various aspects of interest in EA modeling. Additionally, this information model is complemented by means for structuring, which can be considered very helpful in reducing the inherent complexity of the modeling subject. Nevertheless, with the emphasis of the approach on the information model, aspects of visual models and their creation are not addressed in the article. Again, we see the approach presented in section 2 as a valuable contribution in the context, actually providing a way for supporting visual modeling based on the EA metamodel provided in [BW05].

Regarding the inconsistency issue between visualizations and the underlying data, an approach to ensure visualization consistency is pursued in [DV02] and especially focuses on aspects of executability. In order to provide an "open visualization framework applicable to metamodel based modeling languages" the issue is approached from the direction of *visual languages (visualization models)*. Pointing out, that many domain specific visualization environments exist, the approach quickly calls to *XML* as a lingua franca for representing the concepts of these languages. Furthermore, information to be visualized is also serialized as XML, such that concepts of transforming between XML document, as e.g. XSLT can be used for visualizing the information. Nevertheless, the article does not encompass a visual language suitable for expressing the aspects of relative positioning, as the application presented in therein concerns petri-nets and their representation as nodes-and-edges.

Targeting EA modeling, an approach using object-oriented models for describing the EA and the visualizations is given in [SAtDL04]. These models are, similar to the approach presented in section 2 connected via transformations. Nevertheless, these transformations are limited to object-to-object transformations, while the links (instances of associations) are not taken into consideration - again leaving out an aspect crucial for modeling the EA. Furthermore, a language for describing the visualizations as outlined in section 2.2, especially concerning relative positioning, is not provided.

## 5 Outlook

In this article, we emphasized on the importance of models of the enterprise architecture. As we outlined, various approaches and information models for this modeling task exist, with no model or approach being prominent and all-embracing. Complementarily, we outlined the importance of visual models of the enterprise architecture to make the information about the EA perceivable. With the absence of the *one* information model for the EA and the need for visual models obviously existing, the approach presented in Section 2 targets to bridge this gap. Utilizing model transformation concepts and providing a flexible model for describing visualizations, our approach can be seen as an extension to the information modeling approaches as presented in Section 4.

The applicability of the model transformation approach is shown in Section 3 by providing details of a prototypic tool implementation, which is able to ensure consistency between the data modeled according to an arbitrary information model and the visualization representing this data. Nevertheless, the prototypic implementation can be seen as a first step towards a visual modeling tool supporting a variety of information models. Concerning the modeling capabilities further extension for the e.g. for semantic-preserving editing of the visualizations as well as for propagating semantic changes in the visualization to the underlying semantic model have to be explored and currently subjected to research at sebis.

## References

- [BEL<sup>+</sup>07] S. Buckl, A.M. Ernst, J. Lankes, K. Schneider, and C.M. Schweda. A Pattern based Approach for constructing Enterprise Architecture Management Information Models. In A. Oberweis, C. Weinhardt, H. Gimpel, A. Koschmider, V. Pankratius, and Schnizler, editors, *Wirtschaftsinformatik 2007*, pages 145–162, Karlsruhe, Germany, 2007. universitätsverlag karlsruhe.
- [BM03] P. Braun and F. Marschall. BOTL - The Bidirectional Object Oriented Transformation Language. <http://wwwbib.informatik.tu-muenchen.de/infberichte/2003/TUM-I0307.pdf> (cited 2007-01-26), 2003.
- [Buc05] S. Buckl. *Modell-basierte Transformationen von Informationsmodellen zum Management von Anwendungslandschaften*. Diploma thesis, Fakultät für Informatik, Technische Universität München, 2005.
- [BW05] C. Braun and R. Winter. MA Comprehensive Enterprise Architecture Metamodel and Its Implementation Using a Metamodeling Platform. In *Enterprise Modelling and Information System Architectures (EMISA)*, pages 64–79, 2005.
- [DV02] P. Domokos and D. Varró. An Open Visualization Framework for Metamodel-Based Modeling Languages. *Electronic Notes in Theoretical Computer Science*, 72(2), 2002.
- [ELSW06] A. Ernst, J. Lankes, C.M. Schweda, and A. Wittenburg. Using Model Transformation for Generating Visualizations from Repository Contents - An Application to Software Cartography. Technical report, Technische Universität München, Chair for Informatics 19 (sebis), Munich, 2006.

- [Fra02] U. Frank. Multi-Perspective Enterprise Modeling (MEMO) - Conceptual Framework and Modeling Languages. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences 35*, pages 1258–1267, 2002.
- [gaLI06] ATLAS group at LINA & INRIA. ATL: Atlas Transformation Language, 2006.
- [Gro07] The Elver Group. Elver Persistence, 2007.
- [Jam05] G. James. Magic Quadrant for Enterprise Architecture Tools, 4Q04, 2005.
- [KO96] M. J. Kraak and F. Ormeling. *Cartography: Visualization of Spatial Data*. Addison Wesley Longman, 1996.
- [Lau07] S. Lauschke. *Automatische Generierung von Softwarekarten: Entwicklung eines Ansatzes zum Layout deklarativ beschriebener Visualisierungen*. Master's thesis, Fakultät für Informatik, Technische Universität München, 2007.
- [LMW05] J. Lankes, M. Matthes, and A. Wittenburg. Softwarekartographie: Systematische Darstellung von Anwendungslandschaften. In *Wirtschaftsinformatik 2005*, Bamberg, Germany, 2005.
- [LW04] K. Langenberg and A. Wegmann. Enterprise Architecture: What Aspects is Current Research Targeting? Technical report, Ecole Polytechnique Fédérale de Lausanne, Laboratory of Systemic Modeling, 2004.
- [MDG<sup>+</sup>04] B. Moore, D. Dean, A. Gerber, G. Wagenknecht, and P. Vanderheyden. Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework. <http://www.redbooks.ibm.com/redbooks/pdfs/sg246302.pdf> (cited 2007-07-04), 2004.
- [MW04] F. Matthes and A. Wittenburg. Softwarekarten zur Visualisierung von Anwendungslandschaften und ihrer Aspekte. Technical report, Technische Universität München, Chair for Informatics 19 (sebis), Munich, 2004.
- [OMG04] OMG. MOF 2.0 Facility and Object Lifecycle Specification, ad/2004-04-02, 2004.
- [OMG05a] OMG. Revised Submission for MOF 2.0 Query/View/Transformation (ptc/05-11-01), 2005.
- [OMG05b] OMG. UML 2.0 Infrastructure Specification (formal/05-07-05), 2005.
- [OMG05c] OMG. Unified Modeling Language: Superstructure, version 2.0 (formal/05-07-04), 2005.
- [OMG06] OMG. Meta Object Facility (MOF) Core Specification, version 2.0 (formal/06-01-01), 2006.
- [SATDL04] M.W.A. Steen, D.H. Akehurst, H. ter Doest, and M.M. Lankhorst. Supporting Viewpoint-Oriented Enterprise Architecture. Technical report, Information Centre of Telematica Instituut AND University of Kent, Enschede, Netherlands & Canterbury, United Kingdom, 2004.
- [seb05] sebis. Enterprise Architecture Management Tool Survey 2005, 2005.
- [vdTLiD<sup>+</sup>04] L. van der Torre, M.M. Lankhorst, H. ter Doest, J. Campschroer, and F. Arbab. Landscape Maps for Enterprise Architectures. Technical report, Information Centre of Telematica Instituut, Enschede, Netherlands, 2004.
- [Wit07] A. Wittenburg. *Softwarekartographie: Modelle und Methoden zur systematischen Visualisierung von Anwendungslandschaften*. Phd thesis (in publication), Fakultät für Informatik, Technische Universität München, 2007.

# Architecture principles – A regulative perspective on enterprise architecture

P. (Patrick) van Bommel<sup>1</sup>, P.G. (Pieter) Buitenhuis<sup>2</sup>,  
S.J.B.A. (Stijn) Hoppenbrouwers<sup>1</sup> and H.A. (Erik) Proper<sup>1</sup>

<sup>1</sup>Institute for Computing and Information Sciences, Radboud University Nijmegen  
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands  
{P.vanBommel, S.Hoppenbrouwers, E.Proper}@cs.ru.nl

<sup>2</sup>Ordina  
Ringwade 1, 3439 LM Nieuwegein, The Netherlands  
Pieter.Buitenhuis@ordina.nl

**Abstract:** Increasingly, organizations make use of enterprise architectures to direct the development of the enterprise as a whole and its IT portfolio in particular. In this paper we investigate the regulative nature of enterprise architecture. We aim to develop a fundamental understanding of the regulative needs that underly an enterprise architecture, and then take these needs as a starting point to arrive at requirements on the language (architecture principles) used to denote enterprise architectures. We furthermore discuss the process of formulating principles as well as their semantics.

## 1 Introduction

Increasingly, organizations make use of enterprise architectures to direct the development of the enterprise as a whole and its IT portfolio in particular [Lo05]. These developments are fuelled by requirements such as the Clinger-Cohan Act in the USA<sup>1</sup>, which force government bodies to provide an IT architecture.

The term architecture has been used in the field of IT since the 1960's. In the early days it was used to refer to the principles underlying the design of computer hardware and operating systems. This led to the use of the term computer architecture. Later, when software applications became larger and larger, Mary Shaw and David Garlan coined the term software architecture [SG96]. This notion of architecture deals with the key design principles underlying software artefacts. In the 1980's and 1990's people became aware that the development of IT (information technology) should be done in tandem with the development of the context in which it was to be used. This led to the identification of the so-called Business/IT alignment problem [PB89, TC93, HV93]. Solving the Business/IT alignment problem requires organisations to align human, organisational, informational and technological aspects of systems. Quite early on, architecture was also introduced

---

<sup>1</sup>[http://www.cio.gov/Documents/it\\_management\\_reform\\_act\\_Feb\\_1996.html](http://www.cio.gov/Documents/it_management_reform_act_Feb_1996.html)

as a means to further alignment, and thus analyse and solve Business/IT alignment problems [Zac87, TC93, Boa99b]. The Business/IT alignment problem requires the alignment of information technology, consisting of software and hardware, to the other ‘technologies’ used in a business. This has led to the use of the term architecture at the enterprise level [BL96, Boa99a, Lo05]. Enterprise architectures typically bring together a business, information, application and technology perspective on (parts of) an enterprise.

Ultimately, enterprise architectures are a means to an end. The Software Engineering Institute from Carnegie Mellon University identifies the following potential uses [BCK98] for architectural descriptions:

- It is a vehicle for communication among stakeholders.
- It captures early design decisions, both functional aspects as well as quality aspects.
- It describes the global structure decided upon in the architecture, also structures further development.
- It is a transferable abstraction of a system.

Many different perspectives exist on what architecture, in an IT context, actually is. Even though some consensus exists, the field of enterprise architecture is still in its infancy. However, the widespread use of enterprise architecture illustrates that organizations feel a profound need to steer their development (including their IT portfolio) and that they look towards enterprise architecture as a means to fulfill this need. When studying the many existing definitions on architecture [BL96, SG96, BCK98, Boa99a, IEE00, TOG04, Lo05, xAF06], one can discern two important perspectives on architecture:

**Regulative perspective** – Architecture is regarded as a prescriptive notion limiting the design freedom with regards to the design of a system. When taking this perspective one will focus on principles, leading to rules/principles limiting designers in their design freedom.

**Designing perspective** – Architectures are actual specifications of high level system designs focussing on ‘architecturally relevant’ design decisions. When taking this perspective, one typically produces architectural models that describe the design of actual system artefacts.

These two perspectives are complementary in that the regulative perspective accommodates for the need to steer and direct developments, whereas the second perspective supports the need to gain insight into an enterprise’s design while also providing guidance to designers of enterprise systems [Lo05].

In this paper, we focus on the regulative perspective. In taking a regulative perspective on enterprise architecture, we are primarily concerned with its ability to steer the over-all enterprise/system development within a large organization (enterprise). A more specific way of expressing this is to state that “*Architecture serves the purpose of constraining design space*” [xAF06]. In most (enterprise) architecture approaches, this constraining/regulating



is done by means of so-called architecture principles [IEE00, TOG04]. The aforementioned Clinger-Cohen act also requires the architecture to be specified in terms of a set of principles. Such architecture principles usually take the form of informal statements such as (taken from [TOG04]):

*Users have access to the data necessary to perform their duties; therefore, data is shared across enterprise functions and organizations.*

According to the TOGAF architecture framework [TOG04], “Principles are general rules and guidelines, intended to be enduring and seldom amended, that inform and support the way in which an organization sets about fulfilling its mission.” Such principles typically address concerns of the key stakeholders within an organization. In the example case, a stakeholder may be highly concerned about the organization’s ability to flexibly deploy their workforce over different work locations.

While several sources attribute a pivotal role to principles, a precise definition of the concept of principles as well as the mechanisms and procedures needed to turn them into an effective regulatory means still lacks. Both IEEE [IEE00] and TOGAF [TOG04] position principles as a means to guide the design and evolution of systems, while xAF [xAF06] essentially equates (enterprise) architecture to a set of principles. In any case, no clear definition of principles and associated mechanisms and procedures are given.

When considering the definitions reported in literature [TC93, IEE00, TOG04, xAF06], and the definitions used by several practitioners, three key perspectives on principles can be discerned:

**Inherent laws** – These are essentially properties of (classes of) a system that can be observed and validated.

Conceptual parallels are the laws of nature, law of requisite variety, laws of social behavior, etc.

**Imposed laws** – Like inherent laws, they are properties of (classes of) a system that can be validated. However, imposed laws also require mechanisms to enforce them.

Imposed laws typically address concerns of stakeholders. Some of these concerns may be raised by emergent laws having a negative impact on the system being designed.

Examples are: societal laws, policies and regulations within organizations, etc.

**Guidelines** – Desired properties that are so concrete that they offer guidelines to make operational behavior fit imposed laws.

For example: “use your car’s cruise control” is an advisable property to abide by that provides guidance in obeying the law concerning maximum speeds on roads.

In this paper we mainly focus on the last two perspectives on principles.

The remainder of this paper is structured as follows. In section 2 we aim to build up a better understanding of the regulative role of enterprise architectures. This is followed

in section 3 by a discussion of requirements on the language of architecture principles, as a means to operationalize the regulative ability of enterprise architectures. Section 4 then continues by discussing the semantics of principles, in other words, their regulative impact. Before concluding, section 5 discusses an approach to the formulation of architecture principles based on practical experiences and some experiments.

## 2 Enterprise architecture as a regulative mechanism

As mentioned in the introduction, this paper takes a regulative perspective on enterprise architecture. This role comes primarily to the fore in the role of architecture principles [IEE00, TOG04, xAF06]. In [xAF06], enterprise architecture is equated to a set of architecture principles, which are to “limit design freedom”, thus regulating the freedom of an enterprise’s designers.

The aim of this section is to briefly investigate the regulative needs that underly an enterprise architecture. When indeed taking the perspective that an enterprise architecture is a regulative means, one must also agree (at least from a rational perspective) that there is some regulative need motivating the use of the means.

Enterprises have stakeholders. For example: owners, sponsors, people working in the enterprise, clients, etc. Let  $S$  be a stakeholder of an enterprise  $E$ , then it is fair to assume that  $S$  has some goals  $\text{Goals}(S)$  which are potentially impacted on by the behaviour of  $E$ . From the perspective of these goals, the enterprise  $E$  has an ideal behaviour. This behaviour can refer to all aspects of an enterprise, be it the operational processes, financial aspects, labour issues, adaptability, etc.

The actual attainment of  $E$ ’s ideal behaviour from the perspective of  $\text{Goals}(S)$  may be influenced by both internal and external factors [BMM06]. These potential ‘impacts’ may spark stakeholder  $S$  into (trying to) regulate enterprise  $E$  and/or its influencers. Needless to say that  $S$  will not be the only stakeholder, and the desires of  $S$  to regulate  $E$  may indeed conflict with the regulatory desires of other stakeholders.

Given some influencer  $F$ , a risk assessment (see also [BMM06]) may show that  $F$  has a potential undesired impact on  $\text{Goals}(S)$ , in other words there is a set of risks  $\text{Risks}(F, S)$  influencer  $F$  poses to the goals of stakeholder  $S$  by potentially influencing  $E$ . Each of these risks can be quantified in terms of its possible impact  $\text{Impact}(R)$  and probability of occurrence  $\text{Probability}(R)$ , with expected impact:  $\text{Impact}(R) \times \text{Probability}(R)$ . Note: the impact *might* be approximated using an amount of money, but ultimately relates to the impact of a stakeholder’s goals.

If the expected impact is high enough, stakeholder  $S$  will be motivated to introduce regulations to prevent  $R$  from occurring. If  $M$  is some (set of) regulation(s) aimed at  $R$ , then its benefit can be assessed by determining:

$$\text{Benefit}(R, M) \triangleq (\text{Impact}(R) \times \text{Probability}(R)) - (\text{Impact}(R|M) \times \text{Probability}(R|M))$$

where  $\text{Impact}(R|M)$  and  $\text{Probability}(R|M)$  represents the possible impact and probability of occurring of risk  $R$  with regulation  $M$  in place. If  $\text{Costs}(M)$  are the costs of

deploying regulation  $M$ , then the actual effectiveness of  $M$  can be thought of as being the ratio:

$$\text{Benefit}(R, M)/\text{Costs}(M)$$

The costs can, again, be expressed in terms of *money*, but should yet again be regarded as a negative impact on the (satisfaction of) goals a stakeholder is striving for.

Admittedly, this cost/benefit framework is as yet far from practical. This is in line with the observation that the regulative role of enterprise architecture is also still in its infancy. At the same time, however, the desire to use enterprise architecture for such purposes is paramount.

In making this kind of risk assessment more explicit, the field of enterprise architecting may borrow from the field of security [RF07], where security risks are assessed and the effectiveness of security regulations are evaluated against these risks. The aim of this paper is not to develop such a risk assessment, although we subscribe to its necessity, but rather to explore requirements on a principles language. In further research, however, we will indeed invest in a more elaborate cost/benefit analysis approach providing rationalization of principles. In doing so, we will start by evaluating principles (and their enforcement mechanisms) in the context of the scenario's underlying the risks identified by the stakeholders, with the aim of assessing the contribution of these principles in terms of reduced impact and/or reduced chances of the risks occurring.

### 3 Requirements on principles

In this section we focus on the goals of, and requirements on, a modelling language for architecture principles. When taking the position that architecture principles embody the regulative role of enterprise architecture, then we can this as a starting point to identify requirements on a language to express architecture principles. In order to arrive at such requirements, it is important to first make explicit what the goals of the language are [HPW05b]. Without these goals it is difficult to pinpoint the requirements for the language in total. On their turn, these requirements are a mandatory input to formulation of modelling concepts in the language and their requirements [HPW05b, PVH05].

In a survey, in terms of a number of in-depth interviews conducted among a number of experienced enterprise architects [Bui07], the following goals were expressed:

**Regulative architecture** – The language should be designed in such a way that it enables an architect formulate a prescriptive/regulative architecture.

**Supportive; not restrictive** – The language should not act as a straitjacket but should rather should architects in formulating the regulative architecture. It is important that the modelling language does not hinder the (creative!) architecting process itself.

**Architect independent** – Since architecture principles need to be communicated among architects, to stakeholders, and system designers, the language as such should not

be specific to one architect. Even though this may sound obvious, it is still common practice among enterprise architects to come up with one's own language. There is, as yet, not much consensus about such a language.

**Approach independent** – A good modelling language can be used in different development approaches. For example, like UML, ER and ORM can be used in system development methods such as SDM, DSDM, RAD and XP. This should also be the case for this an architecture principles language; it should be independent of the architecting approach used.

**A means of communicating and steering** – Architecture is positioned as a communication and a steering device. This must be taken in consideration when designing this modelling language. The steering and communication goals may lead to conflicts. For example, a nice marketing line (such as Nokia's '*Connecting People*') may be suitable to communicate a basic idea, while not being specific enough to give enough steering.

Based on the above goals, and also as part of the survey, the following requirements were deduced [Bui07]:

**Facilitating role** – The architecting process and not the modelling language should have the most impact on the architecture itself. The architect should not be (too) restricted by the language in formulating the architecture. This is why the modelling language should give the architect some tools and means (in the form of formulation guidelines) to formulate an architecture better. It is then the choice of the architect to use those guidelines. Because an architecture is subject to evolution, it is necessary for the language to be able to support the different stages (from sketch to definitive formulation) of the architecting process.

**Syntactically complete** – The modelling language needs to contain all the different concepts that are used by architects when formulating a regulative architecture. This implies that it must be clear which concepts are used by architects and how they relate to each other. Because each architecture should always be considered in its context, those concepts should be used in the language as well. It is clear that an incomplete modelling language can not produce a complete architecture. The completeness of concepts in the language should be resolved on syntactical level.

**Contains reference architecture** Some architects have pleaded for a modelling language involving numerous examples. This has two advantages:

- it gives the architect possible means to formulate a better architecture and
- it will be possible to create a reference architecture.

Reference architectures play a role of importance in giving the field of enterprise architecting a more mature status. It is still common practice for architects to start from scratch for each new project. This implies that architectures are very often not proven in practice which does not give the client the guarantee that the architecture will work in practise.

**Contains also (semi-)formalised language** – Regulative architectures are currently primarily based on natural language. Interviewed architects do see the advantages of a (semi-)formalised architecture, but claim as well that the architecture should also be communicated to the ‘normal’ stakeholders. The modelling language should therefore not only facilitate (semi-)formalised language, but also the ‘normal’ natural language. The architect should not be forced to write formalised statements.

The formalised concepts can be used by the architect to make a check on completeness and (formalised) linguistic aspects. Because formalised statements are less ambiguous, it’s very advisable to use them with project members who can handle those statements.

When using natural language, there is also a distinction to be made in the degree of abstractness in the formulation. A statement for higher management will normally be different (more concise, more simple, more catchy) than for a project member.

The language is then capable of making different visualizations of the same architecture, focused on the type of stakeholder. Formulating an architecture on a formalised and non formalised level is also consistent with the two main goals of architecture. A (semi-)formalised language supports primarily the steering function, the natural language the communication aspect.

**Terminological framework; improving the communication** – To improve the communication and decrease the ambiguity of the architecture, it’s very important to have a shared term framework between all stakeholders. Such a framework should be based and focused on the stakeholders. However, it is now impossible to insert a fixed term framework in this modelling language because of the architect independence goal. This is also consistent with the requirement that the modelling language should not be too prescriptive.

In addition to these requirements voiced by practitioners, additional requirements can be found in literature. In their Architecture Framework (TOGAF), the Open Group [TOG04] lists five criteria (which are also based on practical experiences) that distinguish a good set of principles:

**Understandable** – The underlying tenets can be quickly grasped and understood by individuals throughout the organization. The intention of the principle is clear and unambiguous, so that violations, whether intentional or not, are minimized.

**Robust** – Enable good quality decisions about architectures and plans to be made, and enforceable policies and standards to be created. Each principle should be sufficiently definitive and precise to support consistent decision making in complex, potentially controversial, situations.

**Complete** – Every potentially important principle governing the management of information and technology for the organization is defined. The principles cover every situation perceived.

**Consistent** – Strict adherence to one principle may require a loose interpretation of another principle. The set of principles must be expressed in a way that allows a balance of interpretations. Principles should not be contradictory to the point where adhering to one principle would violate the spirit of another. Every word in a principle statement should be carefully chosen to allow consistent yet flexible interpretation.

**Stable** – Principles should be enduring, yet able to accommodate changes. An amendment process should be established for adding, removing, or altering principles after they are ratified initially.

The above requirements are requirements on a set of principles and are not requirements on the modelling language. However, these requirements on good principles do underline the need for:

1. a clear semantics of principles, enabling a.o. consistency checks of sets of principles,
2. have a syntax which is understandable by domain experts and not just architects and engineers.

The TOGAF requirements also imply requirements on the process of formulating and maintaining sets of architecture principles.

A further source of requirements on architecture principles and/or a language for modelling architecture principles can be found in the business rules manifesto [Ros03]. Business rules are also forms of regulations that should both have a precise meaning while also be understandable to domain experts/stakeholders. The business rules manifesto lists a set of requirements / principles that should be met by business rules and their application. Most of these requirements also apply to architecture principles. Some key (from an architecture principle perspective) requirements from the business rules manifesto are:

- 3.2 Terms express business concepts; facts make assertions about these concepts; rules constrain and support these facts.
- 3.3 Rules must be explicit. No rule is ever assumed about any concept or fact.
- 4.1 Rules should be expressed declaratively in natural-language sentences for the business audience.
- 5.1 Business rules should be expressed in such a way that they can be validated for correctness by business people.
- 5.2 Business rules should be expressed in such a way that they can be verified against each other for consistency.
- 5.3 Formal logics, such as predicate logic, are fundamental to well-formed expression of rules in business terms, as well as to the technologies that implement business rules.
- 7.1 Rules define the boundary between acceptable and unacceptable business activity.
- 8.4 “More rules” is not better. Usually fewer “good rules” is better.

Most of these requirements are in line with the requirements put forward by TOGAF.

## 4 Semantics of principles

The TOGAF (as well as business rules manifesto) requirement of rules being *consistent* and at the same time being *understandable* by domain experts and stakeholders, provides an interesting challenge in the construction of a principles modelling language.

In [BHPW06, CJN<sup>+</sup>07], we have studied the use of a semi-formal language to represent principles. The language used stems from the field of information modelling, where languages such as ConQuer, Lisa-D and RIDL [Mee82, HPW93, Pro94, BH96, HPW05a] have been used to formulate constraints, rules and queries and reason about these.

Consider as an example the following TOGAF principle:

**Common use applications** – “Development of applications used across the enterprise is preferred over the development of duplicate applications which are only provided to a particular organization.”

A domain analysis and formalization leads to:

**If an** Application *A* [**that** is used in **an** Organization *O* ] results from **some** Development, **and this** Application *A* **is not** a duplicate of **another** Application

[ **that** is used in **another** Organization **than** *O* ], **then that** Development

is preferred by **the** Enterprise **that** includes **both** Organizations and **both** Applications.

The **boldface** words are the keywords of the language, while the non-boldface are domain specific words.

An important question in this example is the way one would have to measure when one application is a duplicate of another application. In making such principles SMART<sup>2</sup>, proper mechanisms should be defined to determine whether one application is a duplicate of another one, or more appropriately, whether one set of applications is a duplicate of another set. And more generally, in the aspect system Business one would like to measure when one process is a duplicate of another process, in order to detect process and organizational redundancy.

## 5 Formulating principles

In [NBP07] we have reported on research efforts into the collaborative formulation of policies/regulations such as architecture principles. This work mainly focussed on the collaborative aspects of such processes. Note that the TOGAF requirements of robustness, completeness and stability of architecture principles have not yet been taken into account in this work. The experience report given in [OP07] did take such requirements into consideration. In [BHPW06, CJN<sup>+</sup>07] the possible effect of using a semi-formal formal modelling language in the formulation of principles was studied.

---

<sup>2</sup>Specific, Measurable, Achievable, Relevant, Time-bound; a common mnemonic used in project management.

Based on these experiments and experiences, we suggest adhering to the following process-structure in formulating architecture principles:

**Assess needs** – An assessments of the regulative needs, which can be used as motivations for the principles to be formulated and their enforcement.

1. (*Collaborative*) In a collaborative session involving stakeholders, sponsors, domain experts and architects, potential regulative needs (issues/concerns/risks) should be gathered. In addition, goals should be formulated upon these regulative needs are to be assessed, and criteria should be formulated regarding the desired longevity of any measures (architecture principles) formulated that are to address these needs.
2. (*Expert-driven*) Risk assessment experts should then assess/judge the identified regulative needs. This assessment should take the form of a risk analysis as suggested in section 2.
3. (*Collaborative*) The stakeholders and the sponsors (of the regulative measures to be taken) should then decide which of these regulative needs they want to see addressed.

**Formulate principles** – The definition of a set of principles that are to be deployed.

1. (*Collaborative*) Based on the (selected) regulative needs, a mix of stakeholders, domain experts and architects should, collaboratively, formulate a set of architecture principles which would address these needs.
2. (*Expert-driven*) The resulting candidate principles should then be pinned down more precisely by a small number of domain experts together with the architects. This group should also assess the longevity of the principles in terms of the criteria produced during the needs assessment, as well as determine more specific consequences of the principle, and clearly identify/quantify the possible contribution of the principle towards the regulative needs.
3. (*Collaborative*) The list of refined principles should then be put to the vote. The domain experts, stakeholders and sponsors should select which principles are to be deployed.

**Prepare deployment** – For each principle a deployment scenario should be formulated.

1. (*Collaborative*) Given the list of selected architecture principles, more refined criteria for the assessment of possible strategies to deploy/enforce these principles should be formulated.
2. (*Expert-driven*) Domain experts and architects should define a number of possible strategies for the deployment/enforcement of the select architecture principles. Each of these strategies should also be evaluated in terms of their costs/benefits.
3. (*Collaborative*) From the list of available scenario's for the deployment of principles, the stakeholders, domain experts and sponsors should select the ones they see as most effective and beneficial.



The above procedure iterates between a collaborative and expert-driven mode. Some tasks should be done collaboratively so as to warrant commitment from, and understanding by, all relevant parties involved. Some other tasks require a focussed effort by a limited number of experts.

Note that the above sketched process structure by no means intends to imply a specific length/duration/size of a specific formulation process. Depending on the requirements of specific situation, such a process may/should require only a single day or even weeks to complete.

## 6 Conclusions and further research

In this paper we have investigated the regulative nature of enterprise architecture. The work reported is part of an ongoing effort to develop a fundamental understanding of the regulative needs that underly an enterprise architecture, and use this understanding in the development of a language for architecture principles as well as a situational procedure/strategy for the formulation of such principles.

In our remaining research activities regarding architecture principles, we identify the following key challenges:

**Language** – How are principles to be expressed, i.e. in what type of language? Are there any hard syntactic or semantic restrictions that are to be imposed, or would this be too restrictive? What could be reasons to (not) impose particular such restrictions? How can understandability be optimally safeguarded at a linguistic level, and how much investment in this is justified in view of quality demands on principle formulations? (How) can SMARTness be increased based on language restrictions?

**Regulative needs** – How can the regulative needs underlying principles be better quantified? How can one predict the positive contributions of enforcing a principle towards these needs?

**Formulation strategies** – Given requirements on the product of the principles creation process, what does such a process look like? Are situational adjustments of the process required or is it possible to define a truly generic process? Apart from the question what semantic and syntactic requirements can be posed for principles, there are pragmatic matters to be addressed. Principles are required to be understood, agreed upon, and committed to by appropriate (groups of) stakeholders. These should be viewed as results of the process, alongside the actual principle formulations [BHP07].

**Deployment strategies** – Enforcement and guidance during the design and development of new systems (and new versions), but also strategies for dealing with legacy systems. How to translate principles and their measuring mechanisms to guidelines? Can the impact of principles be estimated reliably beforehand?

## References

- [BCK98] L. Bass, P.C. Clements, and R. Kazman. *Software Architecture in Practice*. Addison Wesley, Reading, Massachusetts, USA, 1998.
- [BH96] A.C. Bloesch and T.A. Halpin. ConQuer: A Conceptual Query Language. In B. Thalheim, editor, *Proceedings of the 15th International Conference on Conceptual Modeling (ER'96), Cottbus, Germany, EU*, volume 1157 of *Lecture Notes in Computer Science*, pages 121–133, Berlin, Germany, EU, October 1996. Springer.
- [BHP07] P. van Bommel, S.J.B.A. Hoppenbrouwers, and H.A. (Erik) Proper. QoMo: A Modelling Process Quality Framework based on SEQUAL. In H.A. (Erik) Proper, T.A. Halpin, and J. Krogstie, editors, *Proceedings of the Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'07), held in conjunction with the 19th Conference on Advanced Information Systems (CAiSE'07), Trondheim, Norway*, pages 118–127. Tapir Academic Press, Trondheim, Norway, 2007.
- [BHPW06] P. van Bommel, S.J.B.A. Hoppenbrouwers, H.A. (Erik) Proper, and Th.P. van der Weide. Giving Meaning to Enterprise Architectures – Architecture Principles with ORM and ORC. In R. Meersman, Z. Tari, and P. Herrero, editors, *On the Move to Meaningful Internet Systems 2006: OTM Workshops – OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, GADA, MIOS+INTEROP, ORM, PhDS, SeBGIS, SWWS, and WOSE 2006*, Lecture Notes in Computer Science, Montpellier, France, EU, October/November 2006. Springer, Berlin, Germany, EU.
- [BL96] M. Blechar and M. Light. Enterprise Information Architectures. Technical Report R-450–131, GartnerGroup – ADM, February 1996.
- [BMM06] BMM Team. Business Motivation Model (BMM) Specification. Technical Report dtc/06–08–03, Object Management Group, Needham, Massachusetts, USA, August 2006.
- [Boa99a] B.H. Boar. *Constructing Blueprints for Enterprise IT architectures*. Wiley, New York, New York, USA, 1999.
- [Boa99b] B.H. Boar. *Practical steps for aligning information technology with business strategies*. Wiley, New York, New York, USA, 1999.
- [Bui07] P.G. Buitenhuis. Fundamenten van het principe (Foundations of principles). Master's thesis, Institute for Computing and Information Sciences, Radboud University Nijmegen, Nijmegen, The Netherlands, EU, March 2007. In Dutch.
- [CJN<sup>+</sup>07] G.J.N.M. Chorus, Y.H.C. Janse, C.J.P. Nellen, S.J.B.A. Hoppenbrouwers, and H.A. (Erik) Proper. Formalizing Architecture Principles using Object–Role Modelling. Technical Report ICIS–R07006, Radboud University Nijmegen, February 2007.
- [HPW93] A.H.M. ter Hofstede, H.A. (Erik) Proper, and Th.P. van der Weide. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7):489–523, October 1993.
- [HPW05a] S.J.B.A. Hoppenbrouwers, H.A. (Erik) Proper, and Th.P. van der Weide. Fact Calculus: Using ORM and Lisa–D to Reason About Domains. In R. Meersman, Z. Tari, and P. Herrero, editors, *On the Move to Meaningful Internet Systems 2005: OTM Workshops – OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, GADA, MIOS+INTEROP, ORM, PhDS, SeBGIS, SWWS, and WOSE 2005*, volume 3762 of *Lecture Notes in Computer Science*, pages 720–729, Agia Napa, Cyprus, EU, October/November 2005. Springer, Berlin, Germany, EU.

- [HPW05b] S.J.B.A. Hoppenbrouwers, H.A. (Erik) Proper, and Th.P. van der Weide. Understanding the Requirements on Modelling Techniques. In O. Pastor and J. Falcao e Cunha, editors, *17th International Conference on Advanced Information Systems Engineering, CAiSE 2005, Porto, Portugal, EU*, volume 3520 of *Lecture Notes in Computer Science*, pages 262–276, Berlin, Germany, EU, June 2005. Springer–Verlag.
- [HV93] J.C. Henderson and N. Venkatraman. Strategic alignment: Leveraging information technology for transforming organizations. *IBM Systems Journal*, 32(1):4–16, 1993.
- [IEE00] Recommended Practice for Architectural Description of Software Intensive Systems. Technical Report IEEE P1471–2000, The Architecture Working Group of the Software Engineering Committee, Standards Department, IEEE, Piscataway, New Jersey, USA, September 2000.
- [Lo05] M.M. Lankhorst and others. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer, Berlin, Germany, EU, 2005.
- [Mee82] R. Meersman. The RIDL Conceptual Language. Technical report, International Centre for Information Analysis Services, Control Data Belgium, Inc., Brussels, Belgium, EU, 1982.
- [NBP07] J. Nabukenya, P. van Bommel, and H.A. (Erik) Proper. Collaborative IT Policy-making as a means of achieving Business-IT Alignment. In B. Pernici and J.A. Gulla, editors, *Proceedings of the Workshop on Business/IT Alignment and Interoperability (BUSIT-AL'07), held in conjunction with the 19th Conference on Advanced Information Systems (CAiSE'07), Trondheim, Norway*, pages 461–468. Tapir Academic Press, Trondheim, Norway, 2007.
- [OP07] M. Op 't Land and H.A. (Erik) Proper. Impact of Principles on Enterprise Engineering. In H. Österle, J. Schelp, and R Winter, editors, *Proceedings of the 15th European Conference on Information Systems*, pages 1965–1976. University of St. Gallen, St. Gallen, Switzerland, June 2007.
- [PB89] M.M. Parker and R.J. Benson. Enterprisewide Information Management: State-of-the-art Strategic Planning. *Journal of Information Systems Management*, (Summer):14–23, 1989.
- [Pro94] H.A. (Erik) Proper. ConQuer–92 – The revised report on the conceptual query language LISA–D. Technical report, Asymetrix Research Laboratory, University of Queensland, Brisbane, Queensland, Australia, 1994.
- [PVH05] H.A. (Erik) Proper, A.A. Verrijn–Stuart, and S.J.B.A. Hoppenbrouwers. Towards Utility–based Selection of Architecture–Modelling Concepts. In S. Hartmann and M. Stumptner, editors, *Proceedings of the Second Asia–Pacific Conference on Conceptual Modelling (APCCM2005), Newcastle, New South Wales, Australia*, volume 42 of *Conferences in Research and Practice in Information Technology Series*, pages 25–36, Sydney, New South Wales, Australia, January 2005. Australian Computer Society.
- [RF07] S. Raval and A. Fichadia. *Risks, Controls and Security – Concepts and Applications*. Wiley, New York, New York, USA, 2007.
- [Ros03] R.G. Ross, editor. *Business Rules Manifesto*. Business Rules Group, November 2003. Version 2.0.
- [SG96] M. Shaw and D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice–Hall, Englewood Cliffs, New Jersey, USA, 1996.

- [TC93] D. Tapscott and A. Caston. *Paradigm Shift – The New Promise of Information Technology*. McGraw–Hill, New York, New York, USA, 1993.
- [TOG04] The Open Group. *TOGAF – The Open Group Architectural Framework*, 2004.
- [xAF06] xAF working group. Extensible Architecture Framework version 1.1 (formal edition). Technical report, 2006.
- [Zac87] J.A. Zachman. A framework for information systems architecture. *IBM Systems Journal*, 26(3), 1987.

# Service Oriented Security Architecture

Cristian Opincaru  
University of the German Armed Forces, Munich  
cristian.opincaru@unibw.de

Gabriela Gheorghe  
Politehnica University of Bucharest  
gabrielagh@gmail.com

**Abstract:** As Service Oriented Architectures (SOA) and Web services are becoming widely deployed, the problematic of security is far from being solved. In an attempt to address this issue, the industry proposed several extensions to the SOAP protocol that currently reached different levels of standardization. However, no architectural guidelines have yet been proposed. In this paper we first outline the security challenges and the specifications that address these challenges and then present our concept - the *Service Oriented Security Architecture, SOSA*. We argue that the different security functions (authentication, authorization, audit, etc.) should be realized as different stand-alone Web services - *security services*. These security services can then be chained together by means of Enterprise Application Integration (EAI) techniques such as message routing on Enterprise Services Buses (ESB). Next, we will present a prototypical implementation of this framework and describe our experiences so far. We show that by distributing the security functions, a more flexible architecture can be designed that would lower the costs associated with implementation, administration and maintenance.

## 1 Introduction

While Web services were designed to lower integration costs and to open new ways of doing business, many organizations are still reluctant to opening their businesses to Web services although standards like SOAP and WSDL are in place for almost a decade. One of the major factors for this is the inadequate understanding of the security risks involved and the false belief that they will have to make costly reinvestments in their security infrastructures [GFMP04].

In an attempt to make Web services a secure ground, OASIS<sup>1</sup> has standardized a number of extensions to SOAP messaging which address different security issues related to Web services. These extensions are WS-Security, WS-Trust, WS-Federation, WS-SecureConversation and WS-Policy (this last one has been submitted for standardization at W3C). In addition to the SOAP extensions, other security specifications can be used in combination with Web services - XACML [OAS05a], SAML [OAS05b] or the Digital Signatures Services [OAS07a] are some examples.

---

<sup>1</sup>Organization for the Advancement of Structured Information Standards - <http://www.oasis-open.org>

These specifications define how security techniques and mechanisms should be applied for individual SOAP messages (encodings, message exchanges, etc), but they do not define architectural guidelines for possible implementations. In this paper we address this issue by proposing an architecture for the realization of Web services security systems: the *Service Oriented Security Architecture, SOSA* . This architecture is based on the now popular Enterprise Services Bus (ESB) architecture. The idea behind it is to build modular security services that address well defined security functions (i.e. authentication, authorization, etc.). Message routing techniques can be then used to combine these *security services* and to develop complex security solutions.

*SOSA* builds on the idea that rather than creating a "fat" security component which is integrated in the application or in the messaging middleware (and therefore not portable and hard to reuse), it is more appropriate to build security into modular services that are platform independent, easier to test and document, and have a high degree of reusability.

The remainder of this paper is structured as follows: Section 2 describes the main security issues that must be addressed in the context of Web services (references to the specifications that address these issues are made where appropriate), Section 3 presents architectural approaches for the implementation of security systems, Section 4 introduces the proposed architecture and describes its most relevant elements - communication between security services, service composition and possible security services, and in Section 5 we comment on some similar approaches. After this we present our prototype implementation, the *SOS!e* framework, in Section 6, make a functional as well as a performance analysis in Section 7 and finally present our conclusions.

## 2 Security Requirements for Web Services

The main security issues to be addressed by Web services, as also discussed in [GFMP04, WSF<sup>+</sup>03, (W304)], are enumerated below. Because Web services are all about interoperability, we provide references to the specifications that address these security issues, where appropriate.

Please remark that, relative to the OSI network stack, Web services are located at the application layer. Therefore, in this paper we are only addressing application layer security; security at the lower layers is out of scope.

**Authentication** The requester might be asked to provide credentials prior to accessing a Web service. Authentication is a key issue, since without knowing the requester's identity, other security functions can not be accomplished - i.e. you cannot charge someone for using a service without knowing who he is. Authentication is addressed by various specifications, most importantly by WS-Security and SAML [OAS05b] (single sign on).

**Authorization** Access to Web services should be restricted based on authorization policies, that is, clear conditions should be stated under which an entity is allowed to access certain Web services. Authorization is addressed in XACML.

**Confidentiality** The information flow between services must be protected. Special thought

should be given to the fact that SOAP messages often pass through multiple servers before reaching their destination. Confidentiality is addressed in XML-Encryption and WS-Security.

**Integrity** The information received by a Web service must be the same with the one sent by the requester. Messages must not be altered along the path. Integrity is addressed in XML-Signature and WS-Security.

**Non-repudiation** The service provider should be able to prove that a requester used a certain Web service (requester non-repudiation) and the requester should be able to prove that the information he has originates from a certain service provider (provider non-repudiation). Non-repudiation is addressed in XML Digital Signature.

**Privacy** Both service requester and service provider should be able to define privacy policies. Both of them should agree on these policies prior to the actual delivery of the service. Privacy is addressed in WS-Policy and WS-SecurityPolicy.

**Audit** User access and behavior should be traced in order to ensure that the established obligations are respected. Audit is enforced by audit guards, that can be both *active* and *passive* [(W304)].

**Trust** Service requester and service providers should be able to determine if they trust one another. Both direct and brokered trust relationships should be taken into consideration. Trust is addressed in WS-Trust.

**Accounting, Charging** These two aspects are not primarily concerned to the security of the system, but are nevertheless tightly coupled with the other security functions described above (i.e. charging requires the service to know the identity of the requester). Most eBusiness applications require a complete A4C<sup>2</sup> system.

### 3 Security Implementation Approaches

When it comes to implementing the previously introduced security functions in the context of Web services, several architectural approaches are possible. These are graphically presented in figure 1 and described in the following.

**Embedded in the Application** In this case the security implementation is coded in the application itself (figure 1A). The developer of the Web service is responsible for writing the code for the security logic. For this task he will probably chose to implement some of the functionality himself, while reusing some code in the form of 3rd party libraries for implementing other security aspects. Example of such libraries include the Java Authentication and Authorization Service (JAAS) and the security features found in Web Services Enhancements (an extension to the Microsoft .NET platform).

Because the communication between the security system and the application is done by

---

<sup>2</sup>A4C is an acronym for Authentication, Authorization, Audit, Accounting and Charging.

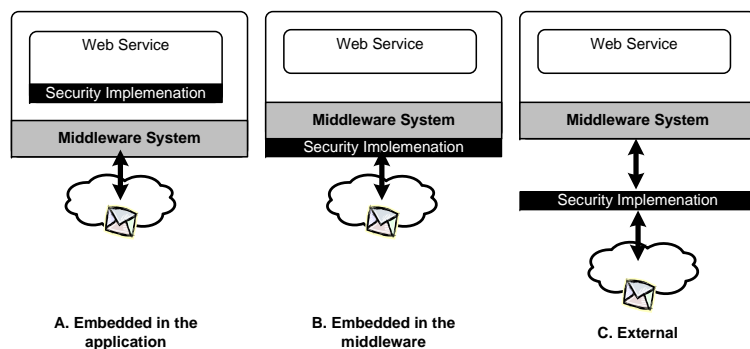


Figure 1: Approaches for the Implementation of Security Features in Web Services

APIs, the performance is very good in this case. However, this approach lacks scalability and results in implementations which are complex, hard to document and have a low degree of reusability and extensibility. These findings are backed by [Bro03].

**Embedded in the Middleware** In this case security is provided by the middleware system where the Web service is executing (figure 1B). This is the case of most application servers such as the Systinet Server for Java<sup>3</sup> or Apache AXIS<sup>4</sup>. Here, the security aspects are handled by the application server. Before and after the Web services hosted by the middleware are invoked, the messages are inspected and the security policies are enforced. In comparison with the previous approach, a noticeable improvement here is the fact that the security implementation is separated from the application logic. This leads to less complex implementations which are easier to document. Furthermore, it is possible to define security policies that cover several services which run inside the same instance of the middleware system. Nevertheless, the security implementation can not be ported on a different middleware system and it is hard to define and enforce policies for services distributed on different middleware systems.

**External** In this case security is implemented outside the middleware system (figure 1C). The Web service is loosely-connected to the security implementation through a messaging interface. This approach is taken for example by XML firewalls - these are deployed at the network perimeter and enforce security policies by processing incoming and outgoing messages. [DGFRLP04] elaborates on application firewalls. In comparison with the previous two approaches, here not only that the security is decoupled from the application, but the two communicate by means of messages. This makes the security implementation independent of the middleware system where the protected Web service runs and results in more understandable implementations (the security aspects are not mixed with the rest of the application). Furthermore, because the security system is

<sup>3</sup><http://www.systinet.com/products/ssj/overview>

<sup>4</sup><http://ws.apache.org/axis>, <http://ws.apache.org/axis2>



essentially a Web service, it has all the advantages that these ones have: scalability, portability, higher degree of reusability.

As disadvantages to this approach we see performance as a potential issue, because there is a significant computational effort associated with message processing, especially if the messages are XML (as is the case of SOAP).

**Mixed** Of course it is possible to have mixed approaches where some security aspects are implemented in the application, some in the middleware, while others are externalized.

## 4 The proposed architecture

In this paper we build on the external security approach described earlier and propose an architecture for security systems where the security functions are realized as small modular services. We call these services *security services*. In order to have a simple, understandable and verifiable design, the principle of separation of concerns is applied. According to this principle, the security system is functionally divided into services and a taxonomy of possible security services will be presented. These services can be regarded as infrastructure services, as they can be shared by applications living in the same network. This makes the design highly reusable. Additionally, through the use of standardized messaging interfaces, overall system portability is ensured.

Enterprise Application Integration (EAI) techniques are used to "glue" the security services together with the Web services which are protected. Because of flexibility, the Enterprise Services Bus model is used. ESBs support both service orchestration and service choreography and implementations usually come equipped with simple-to-use orchestration editors and runtime environments which can easily be used to architect a security solution from security services. Perhaps two of the most important features of an ESB are message routing and the mediation pattern, which allow functionality to be built in the system in a totally transparent fashion.

In this paper, we consider that security services are realized as ESB mediations and that they are chained together by means of message routing. Other possibilities exist (such as for example BPEL or WS-CDL), however these are out of the scope of this paper. Mediations and message routing are enough to design scalable, extensible and easily configurable security systems.

The realization of such a system is illustrated in figure 2. A message reaching the endpoint will be routed by the ESB through several security services before reaching the protected service. Each of these security services implements some security function and will enforce some portion of the security policy. The response message is also routed through several security services before being returned to the requester.

In the proposed model, we consider that the security services trust one-another and that they are located in a trusted network; scenarios such as service hijacking are out of the paper's scope. Nevertheless, by using encryption and digital signatures, the model can be extended to include scenarios where the security services are only partially trusted (they are trusted to perform their task, but not trusted for anything else). However, this is not

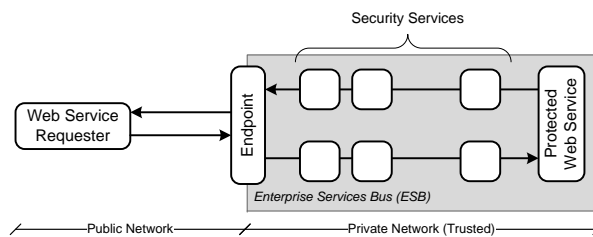


Figure 2: Security system composed of several security services

discussed here.

Because this model is applied to Web services and Service Oriented Architectures (SOA) and because the core idea is to think of security in terms of reusable services, the model was named Service Oriented Security Architecture or *SOSA*. The following subsections present the main elements of this model, namely how security services communicate, how they are coupled together and what security functions can be implemented as services.

#### 4.1 Communication between security services

Each of the security services will process incoming messages in order to accomplish its task. Some tasks may require several services to cooperatively process one message (for example authorization normally requires the identification of the requester). It is clear that in this case intermediary processing results (in the previous example, the identity of the user) need to be exchanged between services.

If we follow the patterns described in [HW03], here are two possibilities for this. The first approach is to have the two services communicate by means of a *shared database*: after processing a message, the first service stores the intermediary results in a database, while the second one later queries this database. The second approach is through *annotations*: the first service appends the intermediary processing results to the message before dispatching it to the next service; we call this an annotated message. For the particular case of security services, this latter approach is more appropriate because the intermediary processing results normally refer to the processed message (i.e. identity attributes, authorization decisions, obligations, accounting information, etc.) and can be therefore transported together with the message.

To have an idea about how annotations work, think about a document-based work flow in a corporation: assume that Bob (an employee) wants a new computer. For this makes a written request and mails it to his boss. The boss will first analyze Bob's reasons, approve it, perhaps annotate it, and forward it to the financial department. The financial department will verify if there are enough funds (perhaps annotate it) and send it to the Infrastructure department. Here the computer is ordered and a reply is sent to Bob informing him that his new computer is on its way. The persons involved in this work flow act similar to the mediation services: first inspect the request they receive, then they approve it (they can

also reject it), they may annotate it, and finally forward it along the chain.

## 4.2 Possible Security Services

In section 2 of this article, the security requirements for Web services were presented. Most of these requirements can be implemented as standalone Web services. In fact, several service interfaces for security services have already been standardized by OASIS as part of the WS-\* specifications. Examples for this include WS-Trust [OAS07b], WS-Federation, XACML [OAS05a] or the newer DSS [OAS07a]. All these services are defined through WSDL documents and follow a request-reply pattern. In this article, as stated earlier, we are looking instead at implementing security as ESB mediation services. Considering this, we argue that the following are candidates for possible security services:

**Authentication** Two types of authentication services are possible: *verification* and *identification*. The first one will verify the credentials (keys, passwords, etc.) found in a message, while the second one is responsible for providing identity attributes. An identification service will annotate messages with these attributes so that the other services along the chain (i.e. authorization, audit, charging) can use this information.

**Authorization** If we follow the XACML [OAS05a] service model, three types of authorization services are possible: *Policy Information Point (PIP)*, *Policy Decision Point (PDP)* and *Policy Enforcement Point (PEP)*. The task of a PIP is to annotate messages with additional attributes that the PDP may require in the decision making process. The task of the PDP is to evaluate the message, produce an authorization decision and annotate the message with this decision and possibly also obligations. The task of the PEP is to enforce the decisions of the PDP services and to discharge the obligations.

**Audit** Two types of audit services can be envisioned according to [(W304)]: services that perform passive audit such as a *logging service* and services that perform active audit such as a *notification service*.

**Cryptographic Services** *Encryption* and *digital signing* are tasks that require significant computational power and therefore ideal for distributing on more powerful machines or machines with specialized hardware.

**Accounting** If accounting represents a complex task, it makes sense to realize it as a standalone service. The task of an *accounting service* is to meter service usage and to provide input for the charging service (in the form of annotations).

**Charging** If charging is done immediately (i.e. not on a periodical basis), the task of the *charging service* is to charge the requester according to the information provided by the accounting service.

**Infrastructure Services** In addition to the above mentioned services, other mediation services might be useful, especially if we think about coupling different security services together. [Cha04] identifies the following three: *orchestration services*, *message transformation services* and *message storage services*.

This list of services is not complete: depending on the concrete deployment scenario, other services may be required. Furthermore, the granularity of the services is also an issue to be considered: concrete implementations may chose to realize several of the up-mentioned services as a single service (for example instead of PIP, PDP and PEP one single authorization service), for performance reasons. Alternatively, in complex systems consisting of different realms, messages may be routed through several PDP services, each one enforcing the policy of its realm. A detailed analysis of these issues is not within the scope of the article. However, in section 7.1 we analyze the performance of our prototype implementation and comment on the relation between the number of security services and message delay.

### 4.3 Putting it all together: Message Routing Patterns

The next design step is to connect the security services with the application Web service. Because the security services are realized as mediation services on an ESB, message routing patterns can be applied in architecting the security solution. Some common patterns applicable here are the following ([HW03] elaborates on message patterns):

**Content-Based Routing** Messages are routed between services based on their content (for example, incoming messages are routed to appropriate identification services depending on the authentication token they contain).

**Itinerary Routing** A routing slip describing the itinerary is attached to the message. This one is then forwarded according to the slip (for example a route may be *authentication - authorization - audit - protected Web service*).

**Splitter / Aggregator** The message flow needs not necessarily be linear. One single request can be split (i.e. forwarded to several services that process it in parallel) and these parallel flows can be synchronized by means of an aggregator which combines the results. Imagine a message being sent in parallel to several decision services and then the authorization decisions being combined by means of AND / OR logic.

In order to illustrate how the proposed architecture fits together, an example is presented in figure 3: after reaching the endpoint, an itinerary is attached to all incoming messages. According to this itinerary messages get first authenticated, then authorized, then logged and only then reach the Protected Web Service. On their way back, response messages are logged, digitally signed and only then returned to the requester. Please remark how services are reused: the same instance of the log service is used twice in the itinerary.

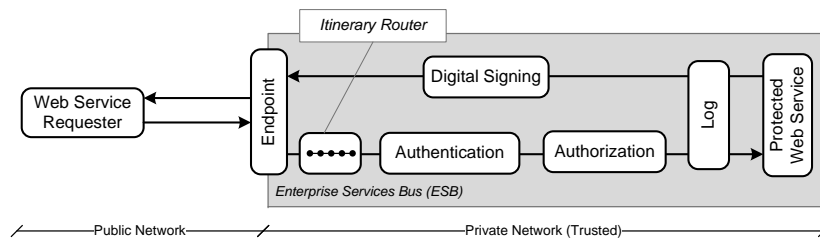


Figure 3: Security Services combined by means of message routing patterns

## 5 Similar Approaches

There are several similar approaches where security functions are implemented as stand-alone services. To begin with, the SOAP protocol specification [(W303)] describes intermediaries which can be either *forwarding intermediaries* (they forward the inbound message with minimal modifications) or *active intermediary* (they modify the outbound message in ways not described in the inbound message). Examples of intermediaries performing security tasks are also given in [(W303)]: a logging service is an example of forwarding intermediary while an encryption service is an example of active intermediary.

In [Bro03] an architecture where security functions are implemented into proxies is described. A single security proxy acting as a gateway is used to secure several Web services deployed in a network. The paper compares this approach to library-based approaches.

[AKT<sup>+</sup>06] enumerates the threats in the context of Web services and describes another external security approach. Here, incoming messages first pass through a *perimeter gateway* which secures several services within a network (similar to [Bro03]), then pass through a *service agent* which is attached to a particular Web service, and only then reach the protected Web service. The security functions are divided in this case between the gateway and the agent: the first one enforces security at a coarser level (for the entire network), while the latter one does it at a finer level (for individual services).

In all these approaches, even though the security functions are separated from the Web service to be protected, they are not realized in a modular fashion. In our approach, the security functions are realized into modular services which are designed according to the principle of separation of concerns - different security functions should be implemented as different security services. The advantages of this approach are presented in section 7.

An approach where security functions are realized into different services is presented in [HHH05]. Here, the services are combined by means of an ESB to form a security gateway. However, only authentication, authorization and cryptographic services are taken into consideration and no communication between security services is designed (in our approach security services communicate by means of annotated messages). Furthermore, no architectural analysis is made, no implementation is presented and hence, neither practical experiences nor performance analysis are described.

## 6 Implementation

In order to show the feasibility to this architecture a prototype implementation was built: the *SOS!e*<sup>5</sup> framework. The implementation is open-source and was realized in Java. It relies on a number of open-source tools, including Apache Tomcat, Apache Axis, Apache Ant, WSS4J, OpenSAML and the Mule ESB. It was designed for SOAP Web services and takes advantage of the SOAP processing model (security services are realized as intermediaries) and several SOAP extensions (most notably WS-Security). The framework implements message routing and the annotation-based processing model described above. On top of *SOS!e* several common security services have been developed.

**Security services** are realized as regular Web services based on the popular Apache Axis platform. The framework provides APIs for the manipulation of annotations. They allow the creation of new annotations, as well as the retrieval, modification and deletion of existing annotations from a message.

**Annotations** have been realized as SAML Attribute Assertions [OAS05b]. These can store several attribute-value pairs together with information about the author of the annotation, timestamp and other fields. SAML assertions have the advantage of being XML encoded, are easy to attach to SOAP message headers (through the WS-Security SAML Token Profile [OAS06]) and have built-in support for digital signatures.

**Message routing** For message routing, the Mule ESB<sup>6</sup> was used. This is a 100% Java based Enterprise Services Bus implementation which supports a variety of transport mediums, message types and routing patterns. It also provides a very convenient and simple way to specify orchestration scripts and to expose these orchestrations as an endpoint. In our implementation (refer to figure 2), a proxy to the protected Web service is exposed in the public network. The Mule ESB is configured to route incoming messages through the necessary security services, before finally invoking the protected Web service. If a request-reply message exchange pattern is used (i.e. the call is not asynchronous) the same happens to the response message.

On top of the *SOS!e* framework, several security services have been prototypically built:

- Two *authentication services* which are able to authenticate users based on username-password and X509 certificates. If the verification is successful, an LDAP repository is contacted, user attributes are retrieved and the message is annotated with these attributes.
- A simple *authorization service* which performs authorization based on simple rules.
- Two *audit services*: one logging service and one alert service. The first one persistently stores messages (in whole or only parts of them), while the latter one can

---

<sup>5</sup>**SOSIE** - **S**ervice **O**riented **S**ecurity, an **I**mplementation **E**xperiment. The name is inspired from the French word *sosie* (look-alike in English), because a proxy of the protected service is exposed through the framework.

<sup>6</sup><http://mulesource.com>

be configured to send emails if certain criteria are met (these are specified through XPath expressions relative to the SOAP envelope).

- Two *cryptographic services*, one for encryption and one for digital signing. The parts of the message to be encrypted / digitally signed are also specified through XPath expressions relative to the SOAP envelope.
- Currently an accounting and a charging service based on PayPal<sup>7</sup> are under development.

## 7 Analysis and Evaluation

It is well known that complexity is security's biggest enemy: as a system becomes more complex, it is harder to observe the flaws and back door opportunities are opened. *SOSA* splits security into small functional components that can be separately developed, thus reducing the complexity and allowing the components to be better tested (unit tests can be used). Because they are less complex, it is also easier to reuse these components.

Services are combined by means of message routing patterns. This allows for a clear design which is also easy to document. Because services are running inside an Enterprise Services Bus, orchestrating the security services is a matter of configuration which does not require an expert, as most ESBs are equipped with graphical editors and specialized tools for such purposes.

Additionally, the fact that the security services are independent one from the other makes the upgrades to the security system faster and less costly. New services can be introduced without affecting the existing ones, by simply altering the path of messages. It is not even necessary to stop the system, the modifications can be done at runtime, by simply temporarily redirecting the message flow (a technique often used when upgrading web servers).

Since the security services are not bound to the application that they guard and also not bound among them, they can be developed in any programming language and can run on any operating system. This will reduce the costs associated with implementation because programmers will be able to choose the APIs and platforms which are most suitable for their project. For example, in an application where user information is stored in Microsoft Active Directory and authorization is based on XACML, the authentication service might be developed in C# (because C# has better support for Active Directory), while the authorization service might be implemented in Java (because Sun offers a free XACML implementation on SourceForge).

Another advantage of this architecture is that the same instance of a service can be used in several applications, thus making the administration and deployment of new services simpler. We showed in figure 3 how the same instance of a security service can be invoked several times in an itinerary. In figure 4 we show how the same security service can be used in two different deployments: in this example the same authorization service is used for both services A and B (the other security services, like the authentication, are different).

---

<sup>7</sup><http://www.paypal.com>

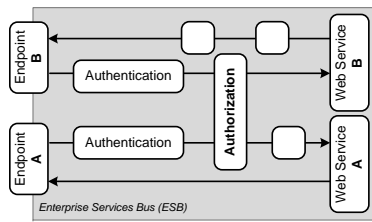


Figure 4: Security service being shared by two security system deployments

Furthermore, sharing security services also solves some well known security issues: sharing the authentication service leads to single-sign-on and sharing the authorization service leads to federated access control.

### 7.1 Performance Analysis

As a possible disadvantage to *SOSA* we see a decrease in throughput and higher latencies due to additional network traffic and overhead resulting from XML parsing (each security service must process the message content). In order to determine the feasibility of *SOSA*, performance testes were carried out against the *SOS!e* prototype implementation.

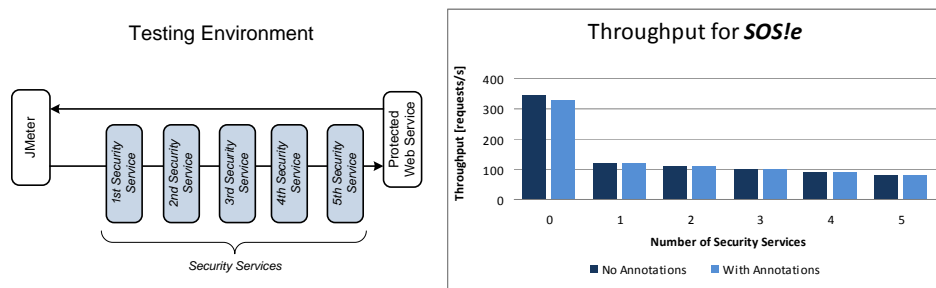


Figure 5: a. Testing environment b. Throughput for the *SOS!e* framework

For tests, mid-class computers were used (Pentium 4 2.8GHz CPU, 2GB RAM, connected via 100MBit network). The results of these tests, together with the testing environment are displayed in figures 5 and 6. We tried to determine the influences of the number of security services on the most relevant performance parameters - the throughput (TP) and the round-trip-time (RTT) for one message.

Our testing methodology is similar to the one described in [UT06]. In order to only measure the overhead introduced by our framework, "dummy" security services were used - these are services that implement no security functionality. The protected Web service, was a very simple one: the purpose was to have this one respond faster than the security



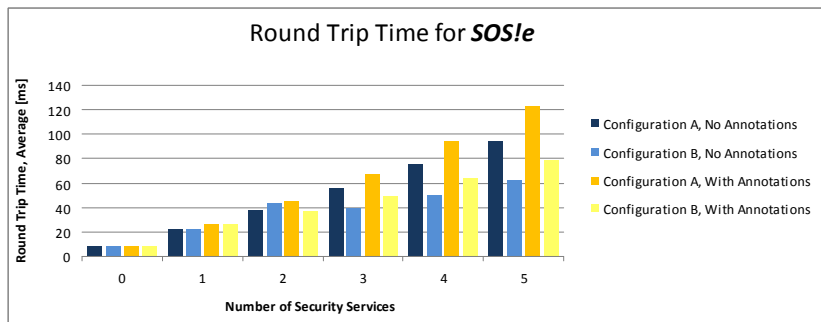


Figure 6: Round Trip Time for the *SOS!e* framework

framework (otherwise this one would have influenced the results).

For the measurements we used Apache JMeter<sup>8</sup>. We considered two different configurations: **Configuration A**, where all the security services were hosted on the same machine as the Mule ESB and **Configuration B** where each security service was hosted on a different machine. For each of these configurations, we tested two use-cases: one where the services were only forwarding the messages and one where the services were processing the annotations existing in the message and adding new annotations.

**Throughput** As seen in figure 5b, the TP decreases significantly when the security framework was introduced between the client and the protected service (almost 60%). This is due to latencies introduced by the Mule middleware. However, if the number of security services is increased, the effect on TP is little. Furthermore there is no difference if annotations are used or not. This shows us that annotations have no visible influence on throughput.

**Round Trip Time** As expected (see figure 6), the RTT increases linearly with the number of security services introduced between the requester and the protected service. The processing of annotations leads to a slight increase in latency.

In conclusion, we see that the framework has significant influence on the performance (both TP and RTT). The decrease in performance increases with the number of security services introduced between the requester and the protected service.

Whether or not the *SOS!e* framework is appropriate for a given scenario depends on the particular performance requirements of this scenario. In those cases where the RTT must be low, the *SOS!e* framework is inappropriate. However, in those cases where the RTT may be higher or if the interactions are asynchronous, *SOS!e* fits well.

Furthermore, we have to take into consideration that *SOS!e* is only a prototype implementation, which is not optimized. Better, more optimized implementations for the proposed architecture can be envisioned.

<sup>8</sup><http://jakarta.apache.org/jmeter/>

## 8 Conclusions

In this paper we presented an architecture for security systems protecting Web services - the Service Oriented Security Architecture. We showed that realizing the security functions into modular, stand-alone security services results in less complex and more flexible designs for security systems. In addition to this, the presented approach has several other advantages (see section 7).

We also presented a prototype, open-source implementation to *SOSA*, the *SOS!e* framework, and showed our experiences with this framework so far. In section 7.1 we presented the results of performance tests that were run on our implementation, and showed that even though both RTT and throughput are affected by the fact that messages are routed through several security services, there are numerous application scenarios in which such an architecture fits well.

## References

- [AKT<sup>+</sup>06] Mohamad Afshar, Nickolaos Kavantzias, Ramana Turlapati, Roger Goudarzi, Barmak Meftah, and Prakash Yamuna. Best Practices for Securing Your SOA: A Holistic Approach. *Java Developer's Journal*, June 2006.
- [Bro03] G. Brose. Securing Web Services with SOAP Security Proxies. *Proc. Int'l Conf. Web Services (ICWS'03)*, pages 231–234, 2003.
- [Cha04] David A. Chappell. *Enterprise Service Bus*. O'Reilly, 2004.
- [DGFRLP04] N. Delessy-Gassant, E.B. Fernandez, S. Rajput, and M.M. Larrondo-Petrie. Patterns for application firewalls. In *Proceedings of the Pattern Languages of Programs (PLoP) Conference*, 2004.
- [GFMP04] C. Gutiérrez, E. Fernández-Medina, and M. Piattini. Web Services Security: is the problem solved? *Information Systems Security*, 13:22–31, 2004.
- [HHH05] Heather Hinton, Maryann Hondo, and Dr. Beth Hutchison. Security patterns within a service-oriented architecture. *IBM white paper*, November 2005.
- [HW03] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2003.
- [OAS05a] OASIS. eXtensible Access Control Markup Language v2.0, February 2005.
- [OAS05b] OASIS. Security Assertions Markup Language (SAML) V2.0 - Core, March 2005.
- [OAS06] OASIS. Web Services Security: SAML Token Profile 1.1, February 2006.
- [OAS07a] OASIS. Digital Signature Service Core Protocols, Elements, and Bindings Version 1.0, February 2007.
- [OAS07b] OASIS. WS-Trust 1.3, March 2007.
- [UT06] K. Ueno and M. Tatsubori. Early Capacity Testing of an Enterprise Service Bus. *Proceedings of the IEEE International Conference on Web Services (ICWS'06)-Volume 00*, pages 709–716, 2006.
- [(W303] World Wide Web Consortium (W3C). SOAP Version 1.2 Part 1: Messaging Framework, June 2003.
- [(W304] World Wide Web Consortium (W3C). Web Services Architecture, February 2004.
- [WSF<sup>+</sup>03] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke. Security for Grid services. *High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on*, pages 48–57, 2003.

# An Approach to use Executable Models for Testing

Michael Soden and Hajo Eichler

Department of Computer Science, Humboldt Universität zu Berlin  
Unter den Linden 6, 10099 Berlin, Germany  
[soden,eichler]@ikv.de

**Abstract.** This paper outlines an approach to test programs by transforming them into executable models. Based on OMG's metamodeling framework MOF in combination with an action language extension for the definition of operational semantics, we use QVT to transform abstract syntax trees as code representations into executable models. We argue that these models provide an adequate abstraction for simulation and testing, since platform dependencies can be resolved in a controlled way during transformation to detach the program logic from its environment. A prototypic implementation based on eclipse EMF underpins the approach.

## 1 Introduction

Execution and simulation of models are well established techniques in software engineering for decades now. While the idea of model-driven architectures (MDA) as proposed by the OMG has been successfully applied to various domains and especially embedded systems, the major part of today's enterprise software systems are *still not* developed in a model-driven way by means of transformations, integrated tool landscapes, rich traceability and 100% code generation. We identified two main reasons for this:

1. Most development languages provide similar abstraction mechanisms than models and come with considerable tool support at the code level
2. Strong execution platform dependencies of the developed code such as library or framework functionality

Worse than this is that the *meaning of models* is typically defined through the mapping into the target environment by code generators. Hence, code generators must be considered to be part of the specification when it comes to (automated) testing between the specification model and the code.

To address these problems, we have created a MOF [1] based framework that supports the definition of *operational semantics* in metamodels to precisely specify the execution semantics of models [2]. Based on the assumption that these metamodels reflect the correct platform behaviour, simulations and tests of the developed code can be executed in the model environment instead of testing it

directly on the target platform. Execution model building is achieved through a transformation defined as QVT relations [3] of a syntax oriented tree metamodel which is close to a language's EBNF grammar (similar to [4]) to an appropriate metamodel for the behaviour definition. Thereby, this import mechanism ensures to reach the proposed abstraction between model and code, which is one of the key ideas of MDA.

## 2 Execution of behavioural models

In order to execute models a (meta-)modelling framework is required that supports the definition and execution of models. For this purpose we use OMG's metamodeling framework MOF [1] in conjunction with OCL [5] and QVT [3] to precisely define and manipulate models in an object oriented way.

Even though MOF defines an overall framework for the definition and management of (meta-)models, it lacks support for the definition of concrete syntax and computational semantics [1]. To fill this gap, we extend the MOF with an action language to support the specification of *operational semantics*[2]. Through the addition of a subset of UML Actions in combination with OCL expressions, the metamodel definitions become machine interpretable and hence models executable. To clearly separate the non-changing model from its runtime configurations which evolve over time, an explicit *instanceOf* relation is introduced at M3. This explicit *instanceOf* modelling reduces any overhead in managing relations between (logical) classifiers and their instances. For this purpose, the *instanceOf* concept is aligned with a create operation which takes care of handling the creation of corresponding links to specified meta-objects.

### 2.1 MOF Actions

We briefly outline the action language in the following along with the sample metamodel of C# used throughout this paper. For a small and complete example refer to [6]. Figure 1 shows a small excerpt of the C# metamodel. The main structural part is conceptually aligned with the UML2 infrastructure library [7], although some minor modifications and simplifications have been applied (e.g. generalization is restricted to single inheritance, some associations are bidirectional, etc.) Rather noteworthy is the addition of language specific concepts such as expressions or delegates. Those parts which are only syntax variations or "syntactic sugar" like property accessors, different kind of loops, etc. are represented by unified concepts in the metamodel.

The operational semantics of the runtime model is described with an action language that is syntactically borrowed from UML Actions/Activities [8]. Figure 2 shows the operational specification of the **CSAssign** meta-class as a sequence of three actions: (1) a *OCL query* retrieving the right hand side of the assignment in the context of the object **self** (which is an instance of **CSAssign**), (2) an *invocation action* that is capable of evaluating the expression and (3) a primitive, single-valued *set* action for the result. Each action is guaranteed to be atomic,

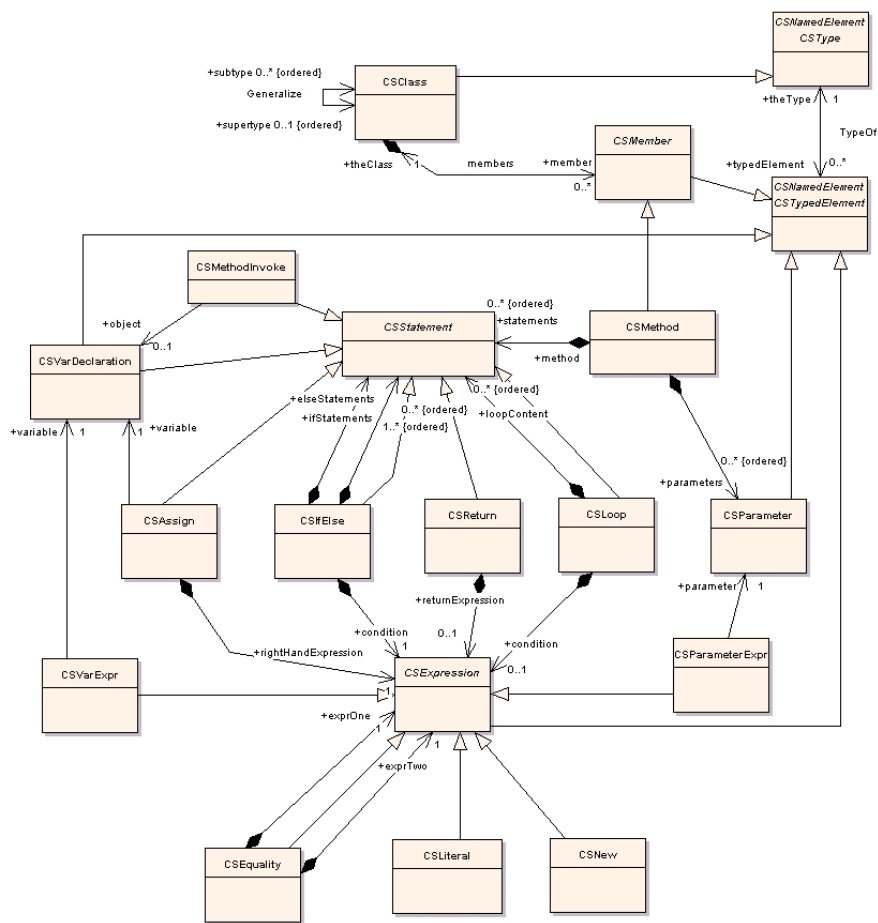
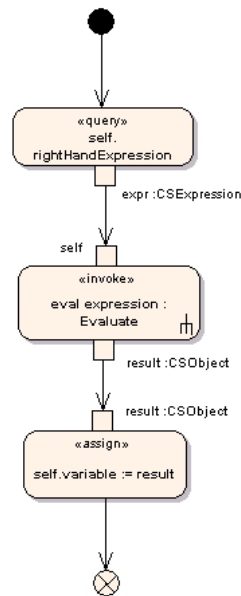


Fig. 1. Excerpt of the C# metamodel: structural parts and expressions

especially queries collecting elements will not be interrupted or interfere with parallel changes applied to the model<sup>1</sup>. Note that `self` in the query and assign actions refers to the owning `CSAssign` object while `self` at the input pin of the invocation action defines the (nested) context for the execution of the `Evaluate` behaviour.



**Fig. 2.** Behaviour of class `CSAssign`

Beside the abstract syntax part of the metamodel, the `C#` runtime model is defined by specific runtime classes (cp. Figure 3). We argue that the runtime model can be regarded as an *instance of* a language’s structural part. For this purpose, the *instanceOf* relationship is introduced at the M3 level to adequately provide support for “logical” multi-metalayer modelling. As consequence, each meta-object has an additional `metaObject` property that points to the specified meta-class. Existing OCL reflection capabilities such as `allInstances` or `oclIsOfType` remain valid and are still bound to the “physical” meta-layering.

Runtime objects can be instantiated with a *create* action. For example, figure 4 (“Create Method Parameter”) shows a behaviour defined in the context of the `CSMethodInvoke` class. It handles allocation and binding of values for all parameters. Note that the type of the input pin of the create action is `CSPParameter`

<sup>1</sup> Hence, there is a global order of all actions executed. Nevertheless, mutual references and modifications to shared objects are allowed

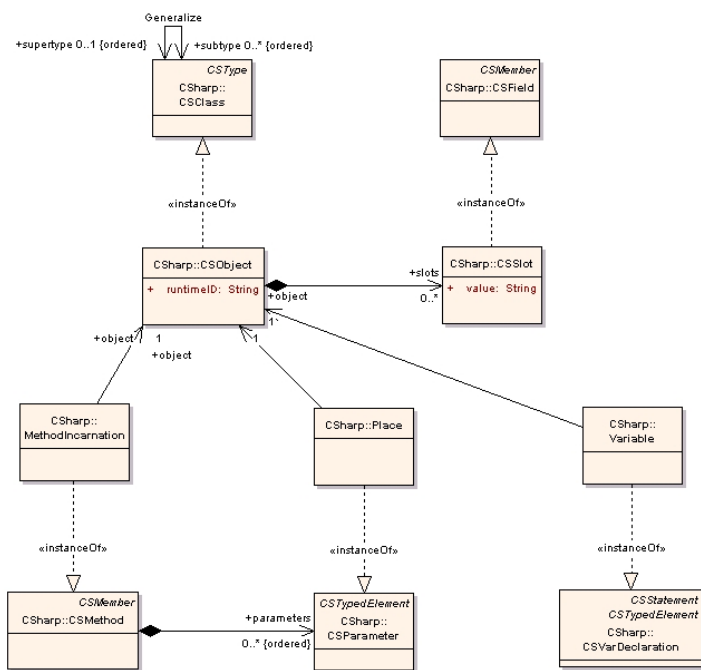


Fig. 3. C# Runtime Model

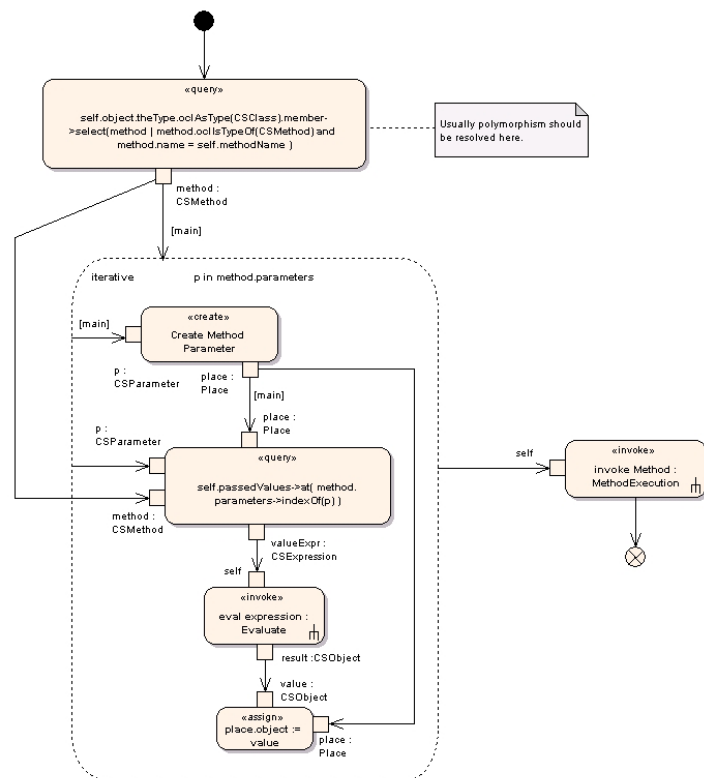


Fig. 4. Actions to specify method invocation



while the output pin is `Place`. Invoking this behaviour causes an object of type `Place` being created as logical instance of class `CSPParameter` with a `metaObject` reference set to the `CSPParameter` object passed to the input pin.

### 3 Execution of code as model

The techniques described above for designing metamodel behaviour are the basis for executing models. Figure 5 outlines the approach to analyse existing implementations in its model representation. The left side of figure 5 outlines the standard MDA approach of model to model transformation. At a certain stage the model is enriched with enough behavioural information to support model execution. The code is transformed into the abstract syntax tree using the generated

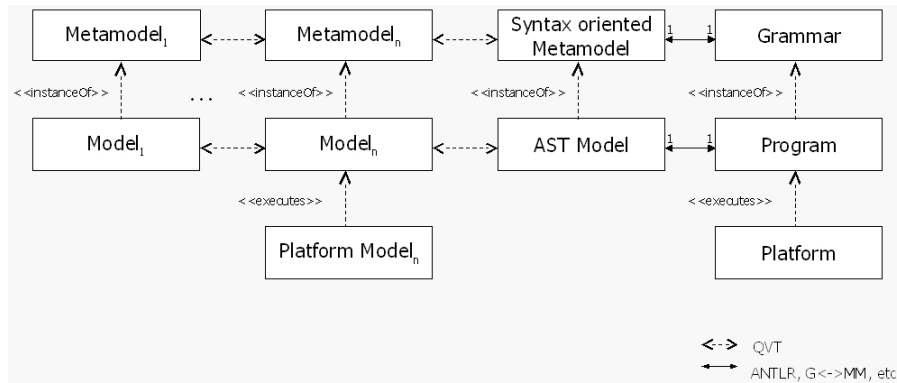


Fig. 5. Mappings between code, grammar and models

parser of ANTLR [9]. This grammar-based representative of the implementation will be mapped to an instance of the syntax oriented metamodel; a one to one mapping to connect the grammar with the modelling elements (compare [4]). The main difference of both representation is the data structure used. Whereas ASTs are defined by a set of independent token types with simple parent/child relation, metamodels offer in addition the advantages of object orientation like inheritance and other modelling techniques like containment. This conversion from grammar to model enables one to apply model transformations on the code representative, but it does not have any effect on the detail degree of the implementation information.

A second mapping transforms the implementation's model into the actual domain specific metamodel. One example of such a metamodel can be found in chapter 2. With this step the goal of abstraction will be achieved by two kind of mechanism. The program itself is abstracted by the mapping to its simplified model. For instance, in the model only one `iterate` definition is defined, whereas

the syntax of the language supports *for*, *while*, *do* etc. loops. The second aspect of abstraction happens by focusing on the program logic itself and extract it from it's surrounding environment. The mapping between the language and the domain specific metamodel is built using QVT relations. The following two examples show a structural and behavioural mapping between the two metamodels. One major reason to use QVT relations here is the possibility for bidirectional transformation that could ensure the re-generation of code from the model in case the model is changed. `Class2Class` as shown in figure 6 defines the mapping of the AST representation of classes to their counterpart in the domain specific metamodel. The patterns of the rule are very elementary to match all occurrences, whereas the classes content is covered by separate rules, which rely to this relation via their *when* clauses as precondition. Large part of the model

```

-- map each class to a class
top relation Class2Class
{
  varName: String;
  enforce domain ast gClass:Class {
    children = qualIdent : QualIdent {
      children = ident : Ident {
        name = varName
      }
    }
  },
  parent = p : TreeNode {}
};
enforce domain metamodel mClass:CClass {
  name = varName,
  scope = namespace : CSNamespace {}
};
when {
  Namespace2Namespace(p, namespace);
}
}

```

**Fig. 6.** QVT rule to map grammar class elements to their model correspondent

transformation forms the behavioural part. One excerpt is the rule *While2Loop*, which expresses the mapping between a *while* control flow statement and the general loop model.

## 4 Related Work

There are many frameworks for model- or language-driven development, development of DSLs, or simulation frameworks with quite different terminology. Metamodelling frameworks or tools include GME[10], XMF[11], Kermet[12], AToM3[13], MetaEdit+[14], AMMA[15] or MPS[16]. As classification of the different approaches by means of support for the definition of structure, static constraints, representation (syntax) and behaviour (execution semantics) as done by Nyttun et.al in [17], we can further distinguish two different approaches to semantics. On the one hand semantics are defined by mappings of models onto

```

relation While2Loop {
  enforce domain ast while:WhileStmt {
    children = exp : Expression (),
    children = gStatements : Statements (),
    parent = p : TreeNode ()
  };
  enforce domain metamodel loop:CSLoop {
    condition = c : CSEExpression (),
    loopContent = mStatements : CSStatement(),
    eContainer = container : CSElement()
  };
  when {
    ContainmentMapping(p, container);
  }
  where {
    Statement2Statement(gStatements, mStatements);
    Expression2Expression(exp, c);
  }
}

```

**Fig. 7.** QVT rule for mapping a while statement to the loop model element

different languages or mathematical formalisms (semantic domains) as in GME and AToM3. On the other hand XMF, Kermet, AMMA and MPS use specific action languages to define operational semantics. The approaches taken in XMF with XOCL (eXtensible Object Command Language), Kermet’s textual action language with OCL and QVT all have in common that querying is achieved by OCL’s navigation capabilities. This idea is reused in our approach. Additionally, the work of [18] inspired us to express the operational semantics with a reduced set of UML actions. However, controlling atomicity of composed actions is rather comparable to the ”step” keyword of the Abstract State Machine Language (AsmL[19]). In the same way as such ASMs define the formal semantics of e.g. the SDL specification[20], our actions follow a similar approach but replace evolving algebras with manipulations of runtime configurations that are instances of MOF metamodels.

Although instantiation is at the core of any metamodeling facility, the approaches differ in their realisation in the frameworks. We argue that while the abstract syntax model is logically at M1, the runtime configurations are located at M0. Atkinson et al.[21] analysed the shallow/deep-instantiation and strict/non-strict metamodeling approaches and pointed out the *ambiguous classification problem* and the *replication of concepts* problem. However, we argue that explicit (shallow) *instanceOf* modelling helps to distinguish multiple logical meta-layers within the concept space defined by a metamodel.

## 5 Discussion

Our approach addresses the problem of decoupled working on model and code level, whereby models do also have a behavioural description of the underlying platform. To execute code as model for testing purposes a couple of advantages against traditional techniques can be found. First, within the abstraction also a major aspect for simplifying testing is found. On the one hand, concentrating on the logic of the program is also a key for writing tests/execute models on the problem scope. On the other hand, it is possible to derive the counterparts of

so called *mock objects*, which emulate a part of the system which is irrelevant for the actual component under test, on importing the code to the model. For example, typical three tier architectures based on the Model View Controller paradigm often requires a lot of code for test-drivers and mock- objects on the GUI and database level. Even though we succeeded only in small replacements of console based input/output, bigger replacements of library functionality could be applied easily. Another important aspect is that execution in the model will lead to scenarios containing model data: the actual testing data. Those scenarios could be recorded and reused for testing.

## 6 Conclusion

The paper outlines an approach to translate existing implementations into their corresponding domain models in order to execute and test their behaviour. The behaviour is defined through an action language extension of MOF that supports the definition of operational semantics. Utilising this approach helps testing the actual implementation by abstracting from the language's concrete environment. Thus, it supports testing and simulation of the implementation decoupled from the platform and other specific library or framework functionality. Our current results are promising that often faced overhead of building test-stubs, simulating network capabilities in test-drivers or omitting GUI references can be solved all at once.

A prototypic implementation has been carried out based on eclipse EMF [22] and a QVT engine [23]. Further directions for the implementation are towards recording of test-runs and comparison against previously executed simulation runs to really test the developed application against its specification.

## References

1. OMG: Meta Object Facility (MOF) 2.0 Core Specification. Object Management Group (2003) ptc/03-10-04.
2. Plotkin, G.: A structural approach to operational semantics. Technical report, University of Aarhus, Denmark (1981)
3. OMG: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, Draft adopted Specification, ptc/05-10-02. Object Management Group (2005)
4. Alanen, M., Porres, I.: A relation between context-free grammars and meta object facility metamodels. Tucs technical report no 606, Turku Centre for Computer Science (2003)
5. OMG: OCL 2.0 Specification. Object Management Group (2005) ptc/2005-06-06.
6. Scheidgen, M., Fischer, J.: Human comprehensible and machine processable specifications of operational semantics. (2007)
7. OMG: UML 2.0 Infrastructure Specification. Object Management Group (2003) ptc/03-09-15.
8. OMG: UML 2.0 Superstructure Specification. Object Management Group (2004) ptc/04-10-02.

9. Parr, T.: (ANTLR – Another tool for language recognition) Last checked: February 8, 2006.
10. Agrawal, A., Karsai, G., Ledeczi, A.: An end-to-end domain-driven software development framework. In: OOPSLA '03: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, New York, NY, USA, ACM Press (2003) 8–15
11. Clark, T., Evans, A., Sammut, P., Willans, J.: Applied Metamodeling, A Foundation for Language Driven Development. Xactium (2004)
12. Team, T.: (Triskell Meta-Modelling Kernel. IRISA, INRIA. [www.kermeta.org](http://www.kermeta.org).)
13. : (The Modelling, Simulation and Design lab (MSDL), School of Computer Science of McGill University Montreal, Quebec, Canada: ATOM3 A Tool for Multi-Formalism Meta-Modelling. <http://atom3.cs.mcgill.ca/index.html>.)
14. MetaCase: (MetaEdit+. <http://www.metacase.com>.)
15. Davide Di Ruscio, Ferric Jouault, I.K.J.B.A.P.: Extending amma for supporting dynamic semantics specifications of dsls. Technical report, Universitegli Studi dell'Aquila (2006)
16. Dmitriev, S.: Language oriented programming: The next programming paradigm. onBoard (1) (2004)
17. Fischer, J., Holz, E., Prinz, A., Scheidgen, M.: Tool-based language development. In: Workshop on Integrated-reliability with Telecommunications and UML Languages. (2004)
18. Sunyé, G., Guennec, A.L., Jézéquel, J.M.: Using uml action semantics for model execution and transformation. *Inf. Syst.* **27**(6) (2002) 445–457
19. Yuri Gurevich, Benjamin Rossman, W.S.: Semantic essence of asml. Technical report, Microsoft Research (2004)
20. ITU-T: SDL formal definition: Dynamic semantics. In: Specification and Description Language (SDL). International Telecommunication Union (2000) Z.100 Annex F3.
21. Atkinson, C., Kühne, T.: The essence of multilevel metamodeling. In: UML'01: Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools. LNCS, London, UK, Springer-Verlag (2001) 19–33
22. Eclipse Project: Eclipse Modeling Framework. (2006) Last checked: January 1, 1970.
23. : medini QVT Engine. ([www.ikv.de](http://www.ikv.de))



# Modelling of Cross-Organizational Business Processes - Current Methods and Standards

Jörg Ziemann<sup>1</sup>, Thomas Matheis<sup>1</sup>, Jörn Freiheit<sup>2</sup>

<sup>1</sup> Institut für Wirtschaftsinformatik im Deutschen Forschungszentrum für Künstliche  
Intelligenz, Saarbrücken

<sup>2</sup> Max Planck Institut für Informatik, Saarbrücken

**Abstract:** Not only since the upcoming of Service-oriented Architectures the modelling of cross-organizational business processes is a heavily investigated field comprising dozens of standards based on different concepts. New techniques on the implementation site, e.g. Web Service orchestration and choreography, further extended the possibilities and requirements on such standards. To systematically order and present a comprehensive state of the art of relevant methods and standards this paper first describes requirements that occur in cross-organizational business processes both for concepts and modelling languages. Then the most important state of the art concepts for modelling cross-organizational processes are described, followed by a list of selected modelling languages. Based on the requirements defined before, a selection of languages is analysed in greater detail.

## 1. Introduction

Enterprises as well as public administrations today are confronted with a highly competitive global and fast changing business environment resulting in an increasing level of cooperation between organizations. This leads to the necessity of implementing interoperable software systems and an efficient modelling of cross-organizational business processes. In the past years research presented various new concepts, standards and tools to support cross-organizational business processes (CBP). The aim of this paper is to provide stakeholders with a comprehensive overview of requirements as well as current concepts and standards for CBP modelling.

A business process is a continuous series of organizational tasks, undertaken for the purpose of creating output. While intra-organizational processes comprise activities executed inside one organization only, the activities comprised in a cross-organizational business process are executed by different organizations that are working together to reach a common objective. Business process models are developed for the purpose of documentation, optimization and automation of business processes. Though models for representing CBPs share these objectives, CBP models differ in various aspects from those for intra-organization processes, e.g. need for information hiding, (higher) need for unambiguous descriptions, need for model usability focused on the collaboration partner and support of flexible relationships.

To explain specifics of CBP modelling and to provide a basis for judging the subsequently described concepts and languages for CBP modelling, section 2 discusses the requirements on collaborative business processes. Besides general requirements, specific requirements on modelling languages are discussed that apply particularly for collaborative business process modelling. Section 3 describes existing concepts that are applicable for CBP modelling. The concepts presented there are also taken from recent literature and research projects as well as tools applied already in industry. In section 4, a selection of currently prominent modelling languages is evaluated both regarding their compliance with the previously described requirements and concepts for modelling of CBPs.

## 2. Requirements for modelling of CBP

In general, modelling languages have to fulfil a set of requirements like flexibility, learnability, good visualization, extensibility, display hierarchy/different levels of granularity, high expressability, executability and analyzability. For an exhaustive list of such generic requirements cp. also Frank und van Laak [FL03]. Based on a literature review (cp. e.g. [WB04]) as well as the results of national and international research projects<sup>1</sup> in the field of cross-organizational business process modelling we deduce the following specific requirements for CBP modelling languages. The requirements will be used later to compare the selected modelling languages.

**Requirement 1: Keep private information private.** Since it can not be expected that each partner publishes its entire workflow and all contained information, this requirement is essential. To allow for publishing the relevant information only, the boundary of the collaboration sphere has to be defined. Various concepts support this, including the specification of static and dynamic system interfaces and the distinction between various levels of privacy/visibility of information.

**Requirement 2: Specify the interfaces of the partners formally.** It is important that the information demand to the partner has to be comprehensive, i.e. that all required information on the interface are sent to the partner. This may comprise the causal relationship at which point in time or after which event information is required, the type of the required information (e.g. document, message) or the number or amount of required information items.

**Requirement 3: Mapping the CBP to executable processes.** The distributed execution of a CBP starts with a common process model that all partners share and that is business oriented. From this model every partner extracts those parts that he has to execute and augments them with arbitrary information he needs for execution, e.g., refinements of process sub-parts or execution context parameters [Gr06]. Thus the used modelling

---

<sup>1</sup> E.g. ATHENA IP (<http://www.athena-ip.org/>), ArKos (<http://www.arkos.info/>), R4eGov IP (<http://www.r4egov.eu/>), Interop NOE (<http://www.interop-noe.org/>).



language should be able to transfer the CBP from business level into an IT-oriented workflow model on technical level like e.g. BPEL.

**Requirement 4: Support of the data flow.** It is important that the data flow of the involved partners of a CBP can be represented by the modelling language [KKS04]. Especially a description of the input needed from partners in order to execute their process parts is necessary.

**Requirement 5: Support of organizational units and roles.** Because different partners are involved in a CBP, it is important to describe the organizational units with the communication and reporting relationships within the CBP. Furthermore, the role concept defines the requirements profile of an organizational unit, particularly necessary for workflow applications. The term “role” describes a certain type of organizational unit with clearly defined qualifications and skills. Thus, the modelling language should be able to describe the different organizational units and roles of the partners within the CBP [KKS04]. The definition of roles also offers to associate activities with roles. Moreover, this association can further be managed by introducing, for example, separation of duties and the management of roles can further be managed by, for example, introducing delegation and revocation of rights and duties.

**Requirement 6: Support of the analysis of the CBP.** Collaborative business process analysis denotes all actions that aim towards measurement and examination of running and finished collaborative processes with the goal of discovering optimization potentials. Once found, such a potential can be realized by changing the process model in the modelling phase of the next cycle pass. Thus, the modelling language should support the analysis of a CBP [MSW06].

**Requirement 7: Support of semantic annotation.** Ontologies are a very popular concept and sometimes commonly used for different purposes. However, here we require a common set (dictionary) of terminologies, a set of relations between terms and their transformations to private processes/terminologies. This includes the possibility of transforming process descriptions (models) from one language into another [HW06]. It also includes the possibility of using the set of terms and their relations for modelling.

### 3. Concepts to support CBP modelling

In the previous section requirements for collaborative modelling have been discussed. In this section we present concepts that have been developed for supporting CBP modelling. Some of these concepts ease the modelling (e.g. interaction patterns), others are directly focused on CBP modelling requirements (e.g. the concept of public, private and global views aims to keep private information private). In Section 4 we then discuss some of the most important modelling languages for CBPs and check both, if they fulfil the requirements presented in Section 2 and which of them support which concepts presented in this section.

An increased competition forces organizations to concentrate on core competencies and to collaborate closely yet flexibly with other organizations. This is true not only for enterprises, but also for public administrations. Thus, methods are required to describe and automate cross-organizational business processes in an efficient manner. In the last decade, the area of cross-organizational processes was investigated intensively, e.g. Van der Aalst<sup>2</sup>, Petri Nets<sup>3</sup>, workflow research<sup>4</sup> and has been by various research Projects, e.g. ArKOS<sup>5</sup>, ATHENA<sup>6</sup> and Interop NOE<sup>7</sup>.

Van der Aalst [Va99] reviewed various forms of interoperability and their usefulness in the context of E-commerce. There he identified the following five forms of realizing interoperable systems by enacting cross-organizational business processes: Capacity sharing, Subcontracting, Chained execution, Case transfer, Loosely coupled. For facilitating the modelling of CBP, van der Aalst also described the Public-To-Private (P2P) approach, which provides the means to specify a common public workflow, to partition it according to the organizations involved and to allow for private refinement of the parts by the organizations, based on a notion of inheritance [VW01]. The P2P approach guarantees that the private workflows of the participating organizations satisfy the public workflow as agreed upon. He also described a top-down (or “outside-in”) approach to come from global processes to private processes.

Schulz and Orlowska’s model is different from 1-tier peer-to-peer model and 2-tier private-public model. They stress that the proposed model framework keeps a minimal set of workflow-relevant data and protocol information, in such a way the workflows can be reused and their privacy be maintained.

Greiner et al.’s work [Gr06] describes the designing and implementing of cross-organizational business processes including different levels of technical detail: the business level, technical level and execution level. They identify how the mappings and transformations are needed among private process, view process and “global” process among the different levels. The business level models illustrate the organizational business aspects as a prerequisite for the successful technical integration of IT systems or their configurations. The technical model derived from the business level model secures the technical realization of the process integration and represents the bridge to the process execution.

Recent research efforts in CBP design were accompanied by the upcoming of new SOA standards and related concepts like Web Service Choreography and Web Service Orchestration. As an outcome, three major types of models supplementing cross-organizational businesses can be observed: First, a “big picture” of the overall CBP, also called global process model, that displays all organizations involved in the CBP and their interactions. Second, models of so called private processes which are executed inside an

---

<sup>2</sup> <http://is.tm.tue.nl/staff/wvdaalst/>

<sup>3</sup> <http://www.informatik.uni-hamburg.de/TGI/PetriNets/>

<sup>4</sup> <http://www.workflow-research.de/>

<sup>5</sup> <http://www.arkos.info/>

<sup>6</sup> <http://www.athena-ip.org/>

<sup>7</sup> <http://www.interop-noe.org/>

individual organization and should not be published (completely) to collaboration partners; nonetheless, they contain some activities that contribute to the CBP. And third, public processes – also called business process stubs - that display only those parts of the private processes relevant for the interaction with the other organizations. The same model types are, however, described differently depending on the research community that uses the model type. For example, Schulz and Orlowska [SO02] also proposed a 3-tier workflow model for cross organizational workflow that captures private partner workflows, workflow-views and coalition workflows. In the following, we shortly describe 7 concepts that in our perception represent the most relevant concepts to be captured in CBP design.

### **Distinguishing between models for private, public and global processes**

As mentioned before, to describe and automate collaborative processes in the last years three different types of process models were introduced [Gr06] [An03]: Private, public and global process models. A **private process** model is described from the viewpoint of an individual organization. Though it may contain activities that represent interactions with other organizations, it is developed for internal use and thus may contain confidential information to be hidden from other organizations. A **public process** model is also described from the viewpoint of an individual organization. It describes the interaction of one organization (e.g. Organization A) with one (B) or more (C) partner organizations. It describes all activities of A being part of this interaction (e.g. “Send RFQ Message to B”, “Receive Quote Message from C”) and the causality of these activities. One way to create a public process is to derive it from a corresponding private process by abstracting all information from it that should be hidden from partner organizations. A **global process** model describes interactions between two or more organizations from a global view point [KL03]. It captures all allowed interactions between all partners involved in the collaboration. Thus, while the public process of A only captures the interactions between the organizations A and B as well as the interactions between A and C, a global process model could contain additionally the interactions between B and C.

While more technical definitions [Bu02] of public processes focus on digital message exchanges, on a more conceptual level interaction models can also describe material exchanges as well as the place and time of such exchanges. A process can be seen as the combination of various organizational dimensions, e.g. the dimensions function, organization, data, output and control [Sc98]. A function represents a business activity, the organizational view describes departments and roles involved in the activity, data and output describe digital and material entities consumed and produced by functions and the control flow combines these views and puts the functions in a timely order. Public processes can be seen as interfaces of private processes and should contain all information necessary to enable the interaction of different private processes. Therefore, besides the sequence of functions contained in an interaction, public processes also have to display information regarding the exchanged data (e.g. which structure an exchanged message has), the goods and services exchanged as well as the organizational departments and roles involved in the interaction.

## **The concept of controllability**

In contrast to public views that are used to offer insight into own private processes, a recent approach has been developed within the concept of controllability [Lo06]. Although this approach has been developed for services (and thus for the execution layer), it can be adapted to the conceptual layer as well. Intuitively, controllability means that a workflow can interact properly with another workflow. In order to detect controllability, a strategy for the own workflow is generated. A strategy describes a set of workflows that could interact with the own workflow. A strategy, usually, is an automaton, which then is sent to the partner. Using this automaton, the partner can check if his own workflow is a proper partner to the other one.

## **Dividing global process models in Swim-lanes**

A swim-lane is a concept for partitioning process models in various subsets, where each subset is executed by one specific actor or organization. Swim-lanes clearly indicate who has the responsibility for carrying out a particular activity or subset of the process. Parallel lines divide the process model into lanes that group activities of the process model by resource definitions, roles, classifiers, organization units or locations. Lanes are arranged either horizontally (rows) or vertically (columns) to divide the process model into logical areas or partitions.<sup>8</sup> Clear distinctions can be made between the processes within an organization unit (within a lane) and those process steps where interactions occur (across lanes). Swim-lanes can contain other swim-lanes which are called child swim-lanes. From a process management perspective, swim-lanes are also used to depict ownership or responsibility of all activities and processes within those swim-lanes. There are a lot of modelling methodologies that use the concept of swim-lanes as a technique to organize activities and to structure the layout of models in order to illustrate different functional capabilities or responsibilities. Swim-lanes are used in UML activity diagrams to logically group activities that correspond to a particular object as well as in the BPMN.<sup>9</sup> Further on, the concept can be used in EPCs in order to divide additional information like data or responsible persons from the current process flow [KKS04]. The use of swim-lanes in the context of SAP Business Scenario Maps<sup>10</sup> aims to indicate how enterprises can collaborate with each other to document the added value potential using a well understandable structure. The swim-lane concept can be used to structure process models. However, the concept does not provide a methodology to model processes. Note that one swim-lane, e.g. the swim-lane for organization A, can also be interpreted as the public process of organization A, since it describes all public interactions that this organization is involved in.

---

<sup>8</sup> Object Management Group (OMG): BPMN 1.0 Specification, 2006, <http://www.bpmn.org>

<sup>9</sup> UML. <http://www.uml.org/>

<sup>10</sup> SAP. Sap business maps. Technical report, SAP AG, <http://www.sap.com/solutions/businessmaps/c-businessmaps/>, 2004.

## **Interaction patterns**

The interaction and communication between different public administrations can be very complex. Even a single communication action within a collaborative process can have a great range of formats, structures and contents. Interaction patterns try to classify messages and to define typical structures based on these classifications. Most of the research that has been done for interaction patterns deals with processes on the execution layer [BDO05] [DP06]. Patterns are used for example to transform BPEL processes into Petri nets in order to analyse the Petri net models and also in order to analyse the controllability of a process.<sup>11</sup> Due to the transformation ability between Petri nets and BPEL, it is possible to use the same patterns that exist for BPEL also for the conceptual layer.

On the conceptual layer there exist several transformations between the different languages, e.g. between EPCs and Petri nets. Service interaction patterns have mainly been developed at the Queensland University by Barros, Dumas and ter Hofstede [BDH05] [BDB05].

## **Visualization of static interfaces**

The trend of cooperation intensifies the need for modelling-methods that consider explicitly the interfaces between more companies participating in one global business process. Generally, an interface, according to the DIN 44300, is defined as an intended or real crossover of the boundary between two units of a same kind respecting the agreed rules for the delivery of data or signals [DIN88]. In object-oriented approaches, the term interface denominates the totality of the public methods of an object [Ha01]. Whereas e.g. the EPC method provides edges and connectors for defining the control-flow, until now, there has been no methodical support for the representation of the crossover between single functions. If two functions that should be processed sequentially reside in two organisations that are separated from each other, the business-process rules that would ensure a smooth transfer of a control-flow from one organisation to another, have to be defined [HK02].

In practice it has been shown that first approaches to model and visualise the cooperative business-processes such as e.g. the SAP AG's C-business maps show a very high aggregation level and simply express the existence of a process-interface without providing a real technical added value. Although the necessity for introducing process interfaces, e.g. in the context of the modelling of the services, has already been detected, a detailed conceptual specification of a transfer from one process partner to another remained undone [KZ03]. A suggestion for a concrete configuration of the conceptual specification of an interface was presented by Kupsch and Klein [KKS04]. In order to get a compact, intuitively understandable visual representation of interfaces in association with e.g. EPC, an interface-diagram is recommended for its conceptual specification. It contains, depending on the company's goals that the entire process

---

<sup>11</sup> <http://sky.fit.qut.edu.au/~dumas/ServiceInteractionPatterns/patterns.html>

supports, substantial dimensions that are necessary for a successful performance of the processes. For interfaces of each collaborative process, an appropriate diagram is created, that is identified through the common name of the process type. The functions that precede or follow an interface make part of a partner-individual pool. In order to ensure transparency, the name of an appropriate function/process module is introduced, depending on the aggregation level applied. Each module can then be specified in various dimensions. Kupsch and Klein propose the dimensions of time, flexibility, place and output for each interaction module [KKS04].

### **Semantic annotation of modelling language**

Many problems associated with CBP are semantic in nature, coming up when describing resources to be exchanged and knowledge to be shared. Hence, if a more automated solution is required, solutions that describe precisely the models of CBPs, resources and information are needed.

In the last years these problems have been studied also in the field of web services to support automatic discovery and composition of services. From this research, a number of results are now available which (partly) are also applicable for CBP modelling. These results are mainly based on the concepts of *reference ontology*, *semantic annotation* and *semantic services*. The basic idea is that resources must be annotated through a reference ontology, i.e. a structured glossary of concepts shared by a community. The annotated resources are stored in repositories and can be discovered through searching algorithms and composed through reconciliation procedures. In general, in CBPs two different aspects of business process models can be identified that are to be annotated semantically: Elements describing the structure of the process model, e.g. control flow elements like logical connectors, and elements describing information, material or other artefacts that are objects used by the process activities, e.g. Business documents, material that is to be sent, money that is to be received, etc. Further on, in the context of CBPs, use of semantic annotations can broadly be categorized into

- Semantic annotation for enabling **horizontal matchmaking**. Horizontal refers to the fact, that the models that are to be matched are on the same level of abstraction. For example, government agency A could provide a semantically annotated EPC model and agency B tries to match its own EPC model with the model of A.
- Semantic annotation for enabling **vertical model transformation** or synchronization. This refers to annotations aiming to describe exactly the elements of a model for connecting it with a model on a different level of (technical) abstraction. For example, the elements of an EPC could be annotated in such a way, that their relation to programming language constructs would be clear.

For the sake of horizontal matchmaking, Thomas and Fellmann [TF06] describe a concept to annotate (graphical) EPC models with graphical OWL models. Correspondingly, they describe how these models can be described with XML representations of both languages, e.g. with EPML and RDF.

## Representing long running transactions

Many real-life CBPs have high requirements regarding consistency and can be running over long periods of time. Especially in distributed environments, it is difficult to control and ensure the consistency of data belonging to one business process. To make this task easier, in recent years the concept of transactionality was taken from the field of database research and was applied to business processes. On the one hand standards for the execution of consistent transactions were created. On the other hand, the need to display secure transactions in business models was met and first modelling languages were offered that contain elements to depict transactions. In general, classical (database) transactions operate through a small period of time and they are characterised by the ACID-principle (Atomicity, Consistency, Isolation und Durability).

It attests that a sequence of activities executed on one system can be seen as a single transaction, if it satisfies following specifications: either all activities are effectually completed or they are eventually not started (Atomicity), the activities cause a consistent system state (Consistency), the activities do not effect any operations that are not part of the transaction, unless this operation is explicitly visible (Isolation), and the sequence is not cancelled by a system error after its execution (Durability) [LR97]. The most popular model for atomic behaviour implementation is the 2-Phase-Commit-Protocol (2PC). However, this includes the locking of the resources affected by the transaction, i.e. the resources are locked at the beginning of a transaction and can not be changed by another transaction until the end of the actual transaction [LR00].

Business process transactions have to be able to cover not transactional programs, long lasting activities and human activities. Such transactions are also called Long-Running Business Transactions (LRT) or Business Transactions. These transactions are supported by the Open Nested Transaction Model and also by the Sagas [GMS87] transaction model [LR97]. Within atomic transaction models (e.g. 2PC) a rollback occurs before the transaction's closure. In case a transaction should not be completed, the locking is taken from the resources. This means that the resources were not changed during the transaction process. However, a compensation action is applied to the transaction activities *after* the transaction's closure, i.e. after the resources have been changed. A compensation sphere is an activities sequence, which either has to be completely executed or completely revised (compensated). For that purpose a compensation activity is assigned either to the single activities or to the whole sphere. This compensation activity is executed, when a transaction has to be rolled back or interrupted [LR97]. Since interoperability should start on the design level and important transactions should be defined by process designers, modelling languages suitable for R4eGov should support advanced transaction mechanisms spanning various parties. BPMN<sup>12</sup> is one of the few languages who include this, and will be presented here as an example. For being able to represent long running business transactions, languages should be able to represent compensation spheres and corresponding compensation activities. The difficulty of this endeavour arises by fact that these elements can appear in various

---

<sup>12</sup> Business Process Modelling Notation Specification, OMG 2006,  
<http://www.bpmn.org/Documents/OMG%20Final%20Adopted%20BPMN%201-0%20Spec%2006-02-01.pdf>.

models and in different levels of detail. For example, the public process interfaces (e.g. 2 different models) of organization A and B can depict elements belonging to the same compensation sphere. Moreover, since transactions can be nested they have to be modelled on various process levels, e.g. in top-processes and underlying sub-processes.

#### 4. Suitability of current standards for CBP modelling

Although there is an abundance of business process modelling languages, only a few were applicable for CBP modelling in practical cases. One major requirement of stakeholders in practice is that business process modelling languages should be widely used in industry and in commercial products. This is the case for EPCs and UML. UML is of additional importance because there is a strong organization behind UML pushing it. This is also the case for BPMN and we have therefore selected it for a more detailed introduction in this section. Another reason of importance is the ability of formal analysis, optimization and verification, which is the case for Petri nets. Thus, in this section we analyse the modelling languages UML, BPMN, EPC and Petri nets in more detail. Note that we selected those for languages from a list of 14 well known standards (cp. [FMZ07]), but due to space restrictions, we omit the other languages here.

If we consider that the requirement to **keep private information private** can be fulfilled by publishing public information of the process only, then this can be done by using a highly abstracted model of the private workflow. “Highly abstracted” in this context means private submodels, i.e. parts of the private workflow, that are just published as, for example, one activity. This is closely related to a hierarchical structure of the model, where on the top level of the hierarchy there is a very abstract presentation of the workflow including all interactions with collaborative workflows and on lower levels there are refinements of this abstraction. However, in order to keep private information private, only the top level needs to be published. The concept of hierarchy can be applied to all four languages. There are in particular broad theories of hierarchy for EPCs and Petri nets. The **formal specification of interfaces** is still a problem in all modelling languages. Petri nets come with formal semantics. The interface of a Petri net model usually is specified by places that act as channels for message passing or document exchange. The types of data to be exchanged can formally be specified using coloured Petri nets. However, even for Petri nets there is no special (graphical) element that models an interface. UML and in particular BPMN and EPC are more powerful in terms of modelling interfaces having special modelling elements for interfaces and triggering events. They, however lack formal semantics. The **mapping to executable processes** is possible for all languages. In particular, mappings from all for modelling languages to BPEL exist. However, this cannot be done purely automatic but with computer support. For BPMN a mapping to BPEL is already contained in the languages specification and BPMN elements are matching well with BPEL concepts, for example both languages support elements for distributed transactions. EPC are also a suitable basis for BPEL transformations (cp. [ZM05]) and the core elements of EPCs (functions and events) map to the core elements of BPEL (web service invocations and various types of events). Petri nets can be executed even without mapping to BPEL due to its formal semantics. The BPMN specification already contains a detailed description of transformation to



BPEL. **Data flow** is fully supported by Petri nets due to their ability of specifying very complex types (for coloured Petri nets) of *tokens*, i.e. the data or information are modelled explicitly and the tokens are evaluated in order to compute the occurrence of certain events. In the other languages data flow is not directly supported (no data flows through the process models) but they offer modelling elements for different types of data (data bases, documents, etc.), which can be associated with activities, such that the flow and the change of data can be modelled indirectly. All languages lack on explicit support of **involved roles**. However, EPCs are suited here because of their ability to model organizations and to specify who is in charge of certain activities. There is no direct support of modelling resources (e.g. staff members) in Petri nets but this can be done by modelling resource places and marking them with corresponding tokens. The ability to **analyze** collaborative business processes requires formal semantics. Petri nets have a formal semantics per definition. There has also been done work on EPCs, UML and BPMN in order to introduce formal semantics to those languages. There has been done work for **semantic annotation** for EPCs. However, to our knowledge there is no other approach on semantic annotation for the other selected languages.

Table 1: Evaluation of selected business process modelling languages regarding the requirements of Section 3<sup>13</sup>

	UML	Petri nets	BPMN	EPC
Keep private information private	x	x	x	x
Specify the interfaces formally	-	o	-	o
Mapping the CBP to executable processes	x	x	x	o
Support of data flow	o	x	o	o
Support of involved roles	-	o	-	x
Support of analysis of the CBP	-	x	-	x
Support of semantic annotation	-	-	-	o

For all four languages the concept of **Swim-lanes** is applicable. Though use of swim lanes is more common for BPMN and UML, they are as well applied to EPC and Petri nets. As discussed above for the requirement of keeping private information private, in general for it is possible for all process languages to derive **global and public processes from private processes**. However, to our knowledge explicit approaches for this kind of transformations exist for EPCs and Petri nets only. Concepts for **visualization of static interfaces** was described for the EPC [KKS04] and makes most sense for business oriented languages, e.g. the more formal Petri Nets are less suitable for such visualizations. Among the four languages, **semantic annotations** of business process modelling languages so far exist only for the EPC [TF06]. **Long running transactions** are supported explicitly only by BPMN. The concept of **controllability** is closely related to the concept of private and public workflows. It has been developed for Petri nets but can certainly be applied to the other languages as well. **Interaction patterns** mainly

<sup>13</sup> Criteria is satisfied (+), criteria is partly considered (o), criteria is not supported (-)

exist for Petri nets and BPEL but can certainly be developed for the other languages as well.

Table 2: Evaluation of selected business process modelling languages regarding the concepts of Section 3

	<b>UML</b>	<b>Petri nets</b>	<b>BPMN</b>	<b>EPC</b>
Swim-lanes	x	o	x	o
Private, public and global processes	o	x	o	x
Visualization of static interfaces	o	-	o	x
Semantic annotation of modelling languages	-	-	x	x
Representing long running transactions	-	-	x	-
Controllability	o	x	o	o
Interaction patterns	o	x	o	o

## 5. Summary and future research

To describe and analyse existing approaches to model CBPs we first described requirements distinct for cross-organizational scenarios. Then state of the art concepts for modelling the conceptual layer of collaborative processes were described, including swim-lanes, semantical enriched processes, interaction patterns, and the distinction between public, private or global views, which are supplemented by the controllability approach. The latter approach, however, is not ready for application yet and has to be tested sufficient for real processes. Moreover, it has been suggested for business process execution and not yet applied to conceptual models. Choosing from a list of 14 well known standards, we selected and analysed EPC, BPMN, UML and Petri Nets based on the gathered requirements. Due to their explicit support of business elements, EPC and BPMN seem to be the most suitable candidates for modelling CBPs. However, for BPMN an established commercial tool is missing, which do exist for EPCs (e.g. the ARIS Toolset). Petri nets are more appropriate for formal analysis of business processes. However, besides the missing commercial tool, there is also a lack of modelling power in terms of different available process elements, such as organizational diagrams, interfaces etc. Nonetheless, Petri nets have implicit formal semantics and are able to model objects flowing through a process. There exist several enhancement and transformation approaches for EPCs, e.g. EPC models can be directly transformed into executable formats, such as BPEL or enriched with semantic annotations. Thus, our future research will concentrate on applying and extending BPMN and EPC for cross-organizational business process modelling.

The work published in this paper is (partly) funded by the E.C. through the R4eGov project. It does not represent the view of E.C. or the R4eGov consortium, and authors are solely responsible for the paper's content.

## Literature

- [An03] Andrews, T., Curbera, F., Dholakia, H., Golland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I. and Weerawarana, S. (2003). Business Process Execution Language for Web Services – Version 1.1. 2003.
- [BDB05] Barros, A., Dumas, M., Bruza, P.: The Move to Web Service Ecosystems. BPTrends Newsletter, Volume 3, Number 12, December 2005.
- [BDH05] Barros, A., Dumas, M., ter Hofstede, A.: Service Interaction Patterns: Towards a Reference Framework for Service-based Business Process Interconnection. Technical Report FIT-TR-2005-02, Faculty of Information Technology, Queensland University of Technology, Brisbane, Australia, March 2005.
- [BDO05] Barros, A., Dumas, M., Oaks, P.: Standards for Web Service Choreography and Orchestration: Issues and Perspectives. In Proceedings of the Workshop on Web Service Choreography and Orchestration for Business Process Management, Nancy, France, Springer Verlag, September 2005.
- [Bu02] Bussler, C., Public Process Inheritance for Business-to-Business Integration. In Buchmann, A. P., Casati, F., Fiege, L. and Shan, M. C.: Technologies for E-Services – Third International Workshop TES 2002, Hong Kong, 2002.
- [DIN88] Deutsches Institut für Normung e.V.: DIN 44300 : Informationsverarbeitung : Begriffe. Teil 1. Berlin : Beuth, 1988.
- [DP06] Decker, G., Puhlmann, F.: Formalizing Service Interactions Extended version of a paper to be published in the 4th International Conference on Business Process Management (BPM'2006), Vienna, Austria, September 2006.
- [FL03] Frank, U., van Laak, B.: Anforderungen an Sprachen zur Modellierung von Geschäftsprozessen. Arbeitsbericht des Instituts für Wirtschafts- und Verwaltungsinformatik der Universität Koblenz. [Http://www.uni-koblenz.de/~iwi/publicfiles/Arbeitsberichte/Nr34.pdf](http://www.uni-koblenz.de/~iwi/publicfiles/Arbeitsberichte/Nr34.pdf), Stand: 26.2.2004.
- [FMZ07] Freiheit, J.; Matheis, T., Ziemann, J.; Definition of static and dynamic models of collaborative workflow interoperability. Deliverable D4.1, R4eGov – Towards e-Administration in the large. IST-2004-026650.
- [GMS87] Garcia-Molina, H., Salem, K.: Sagas ; in ACM SIGMOD; 1987; San Francisco; pp. 249-260, 1987.
- [Gr06] Greiner, U., Lippe, S., Kahl, T., Ziemann, J., Jäkel, F.W.: Designing and implementing cross-organizational business processes - description and application of a modelling frame- work. In Proceedings of the Interoperability for Enterprise. Software and Applications Conference (I-ESA 2006).
- [Ha01] Hansen, H. R.: Wirtschaftsinformatik I. 8. Auflage. Stuttgart : Lucius & Lucius, 2001.
- [HK02] Herrmann, K.; Klein, R.: Effizientes Schnittstellenmanagement : Erfolgsfaktor für die E-Collaboration. In: IM – Information Management & Consulting Nr. 4, 2002.
- [HW06] Herborn, T., Wimmer, M.: Process Ontologies Facilitating Interoperability in eGovernment - A Methodological Framework. In: Proceeding of the Workshop on Semantics for Business Process Management. p.76-88, 2006.

- [KKS04] Klein, R., Kupsch, F., Scheer, A.-W.: Modellierung inter-organisationaler Prozesse mit Ereignisgesteuerten Prozessketten. In: Scheer, A.-W. (ed.): Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 178, Saarbruecken, 2004.
- [KL03] Khalaf, R., Leymann, F.: On Web Services Aggregation. In: Benatallah, B., Shan, M. (eds.): Technologies for E-Services. Lecture Notes in Computer Sciences 2819, Springer, Heidelberg, 2003.
- [KZ03] Klein, C.; Zürn, A.: Einsatz von Prozessmodulen im Service Engineering : Praxisbeispiel und Problemfelder. In: Bullinger, H.-J.; Scheer, A.-W. (Hrsg.): Service Engineering : Entwicklung und Gestaltung innovativer Dienstleistungen. Berlin [u. a.]: Springer, 2003, pp. 737.
- [LR00] Leymann, F.; Roller, D.: Production Workflow - Concepts and Techniques PTR Prentice Hall, 2000.
- [LR97] Leymann, F.; Roller, D.: Workflow based applications, IBM Systems Journal 36(1) pp. 102-123, 1997.
- [Lo06] Lohmann, N., Massuthe P., Stahl Ch. And Weinberg D.: Analyzing Interacting BPEL Processes. Intern. Conf. on Business Process Modelling (BPM 2006). 2006
- [MSW06] Matheis, T., Simon, B., Werth, D.: Process-Based Performance Measurement of Networked Businesses. In: Cunningham, P.: Cunningham, M. (Hrsg.): Exploiting the Knowledge Economy - Issues, Applications, Case Studies, eChallenges e-2006 Conference, Barcelona, 25.-27, pp. 20-28. (ISBN: 1-58603-682-3), Oktober 2006.
- [Pe62] Petri, C.A.: Kommunikation mit Automaten. Dissertation, Technische Universität Darmstadt. Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.
- [Sc98] Scheer, A.-W., ARIS – Vom Geschäftsprozess zum Anwendungssystem. 3. edition. Springer, Berlin, 1998.
- [SO02] Schultz, K., Orłowska, M.: Towards a cross-organizational workflow model, Proc. 3rd IFIP Conf. on Infrastructures for Virtual Enterprise, May 1-3, 2002, Sesimbra, Kluwer.
- [TF06] Thomas, Oliver; Fellmann, Michael: Semantische Ereignisgesteuerte Prozessketten. In: Schelp, Joachim; Winter, Robert; Frank, Ulrich; Rieger, Bodo; Turowski, Klaus (Hrsg.): Integration, Informationslogistik und Architektur : DW2006, 21.-22. Sept. 2006, Friedrichshafen : Proceedings. Bonn : Köllen, 2006 (LNI, P-90), S. 205-224.
- [Va99] Van Der Aalst: Process oriented architectures for electronic commerce and interorganizational workflow. Information Systems,24(8):639 – 671, 1999.
- [VW01] Van der Aalst W.M.P., Weske, M.: The P2P Approach to Interorganizational Workflows, Proceedings of the 13th International Conference on Advanced Information Systems Engineering, p.140-156, 2001.
- [WA04] Wombacher, A.; Aberer, K.: Requirements for Workflow Modeling in P2P-Workflows derived from Collaboration Establishment; in Proc. 1st Intl. Workshop on Business Process Integration and Management (BPIM 04), Zaragoza, Spain, IEEE Computer Society Press, ISBN: 0-7695-2195-9, p 1036-1041, 2004.
- [ZM05] Ziemann, J.; Mendling, J.: Transformation of EPCs to BPEL – A pragmatic approach. 7th International Conference on the Modern Information Technology in the Innovation Processes of the industrial enterprises, Genoa, Italy, 2005.

# Using BPEL as a workflow engine for local enterprise applications

Nicolas Biri, Pascal Bauler, Fernand Feltz, Nicolas Médoc, Céline Thomase

Centre de Recherche Public – Gabriel Lippmann  
rue du Brill 41,  
4422 Belvaux  
Luxembourg  
{biri, bauler, feltz, medoc, thomase}@lippmann.lu

**Abstract:** This paper gives an overview on the integration of a BPEL workflow engine into an enterprise application in order to decouple business processes and application code. The technical complexity of this innovative approach is hidden by means of Model Driven Software Development (MDS) techniques and several component frameworks. By referring to a research project realised in collaboration of the Centre de Recherche Public – Gabriel Lippmann and the Luxembourg National Family Benefits Fund (CNPF), with the overall goal to optimise the IT environment of the CNPF, this paper shows how the proposed approach is particularly adapted to agile and iterative development projects.

## 1 Introduction

The project presented in this paper is realised in collaboration with the Luxembourg National Family Benefits Fund of Luxembourg (CNPF (Caisse Nationale des Prestations Familiales)). This administration is in charge of the payment of family allowances for people working in Luxembourg. During the last few years, the CNPF has started a modernisation and optimisation process highly relying on information technologies. There are mainly two reasons for this process: First of all, the modernisation is mandatory to enable the handling of the growing workload and the complexity of the underlying treatments, which mainly result from the increasing number of borderline commuters and the extension of the European Union. A second reason for this modernisation effort consists in offering adequate e-government solutions improving the quality of service offered to the citizens and so increasing the interest of the population in modern computer technologies. Due to the geographical situation of Luxembourg and the high number of borderline commuters, complex cross-administration and cross-country solutions have to be designed and data exchange protocols have to be specified, in order to enable the access to the heterogeneous IT environments of the different neighbouring countries. Strategic investment in modern IT solutions is justified, as the workforce of the CNPF basically remained unchanged, although the number of commuters significantly increased over the last decade. After some preliminary and

internal discussions, the Centre de Recherche Public – Gabriel Lippmann got involved in the modernisation process [Ba06, HF06], to design innovative solutions. In this paper we present a part of this modernisation project by mainly focusing onto the design of enterprise solutions handling the computations of family allowances for commuters. We show how modelling technologies combined with agile management concepts, significantly help to successfully accomplish this project and to move into production.

Considering the short development phases and the numerous changes required in the IT environment of the CNPF, we heavily rely on the SCRUM concepts [SB01] to drive the project. This approach is well adapted for projects with short deadlines running in an evolving context. The key principle of this method inspired by the agile development method, is to define priorities on the requirements and to establish short incremental development cycles (called *sprints*), each of these containing a few goals to achieve (called *backlogs*). Several operational and functional tests have to be passed before a development cycle can be closed. Furthermore, a development cycle is usually ended by a practical demonstration involving the partner and the end-user. In the particular context of the CNPF we consider 2 types of demonstrations, either showing newly designed business functionalities or discussing the progress in the specification protocol with the neighbouring countries. This differentiation between operational and specification results, is due to the need to collaborate with foreign development teams often relying on the Waterfall model.

To take maximum advantage of the iterative development cycles introduced by the SCRUM approach, the used technologies and development framework are selected based on their compatibility with iterative development. This stresses the decision to realise the application flow by means of executable workflows, in addition Model Driven Architecture (MDA) and Model Driven Software Development (MDSO) are used to generate technical and repeatable code segments and as a consequence, to improve the overall development process.

From a technical point of view, a BPEL (Business Process Execution Language) workflow engine is introduced to coordinate the core workflows and business processes of the proposed solution. The common usage of BPEL engines consists in orchestrating services (usually based on web-services) including handling of incoming messages, message transformation and message routing. BPEL engines offer by default a message centric approach, where the analysis of incoming messages determines further treatments, either by initiating new processes or by passing progress information to existing processes. Introducing appropriate MDSO frameworks, which hide technical aspects of the overall solution, facilitates efficient usage of BPEL. The design decision to build enterprise application architectures around a BPEL engine is justified as follows:

- The family allowances business domain is frequently adapted to political decisions and legal changes, which results in regular changes of the underlying business processes. By decoupling application code and application workflows, maintenance and enhancement aspects can be optimised.

- Furthermore, decoupling of application workflows and code is especially adapted to agile and incremental development projects. The IT teams can start with simplified workflow skeletons, which are systematically enhanced and adapted during the various development cycles of the project.

Below, these aspects are discussed in more detail. Section 2 presents the project context with an overview of the proposed solution. In section 3, the orchestration solutions and more precisely the BPEL specificities are exposed. The advantages and issues resulting from the use of BPEL as a workflow engine in an incremental development process, and its integration into our solution, are explained. This section also discusses some technical aspects required to avoid de-synchronisation between BPEL workflows and the application code. Section 4 explains how MDA technologies (Model Driven Architecture) facilitate this synchronisation and how this technology fits with the agile approach. Section 5 concludes this paper.

## **2 Project overview**

### **2.1 Project working plan**

The general project goal is to automate the computation of the family allowances for the people working in Luxembourg. We distinguish the family allowances for Luxembourg citizens on one side and for commuters on the other side. The complexity of the family allowances results from a European decision saying that family allowances are exportable. So each person working in Luxembourg, independently of his or her residence country, is granted the Luxembourg family allowances. Furthermore the citizens get the allowances from where they are the highest, either from the residence country or from the working location. As in Luxembourg the allowances are higher than in the neighbouring countries, the practical situation is somewhat simplified. As a consequence, each family with incomes resulting exclusively from activities in Luxembourg is treated as resident in Luxembourg. The situation is more complex for families with incomes resulting from activities in different countries. The current procedure consists in having the residence country pay the family allowances on a monthly basis. The difference between the local and the Luxembourg's allowances are calculated twice a year and are directly paid to the citizen. As this process is error prone and tedious, a first project goal is to replace this process in order to pay the allowances on a monthly basis and to delegate eventual clearing operations to the back-end IT systems. This improvement however requires an excellent collaboration between the Family Allowances Funds of the neighbouring countries. Special political agreements have to be established before facing technical burdens related to the heterogeneous IT environments. These technical issues are discussed in detail in the following paragraph.

Due to the high increase of the number of commuters in Luxembourg, the CNPF noticed in 2001 that they were no longer able to handle all the files manually. At that time, roughly half of the commuters came from France. That justified the decision to tackle the French commuters in priority and to start discussions with French allowances offices. An interesting factor was that the French Family Benefits Fund is organised in a semi-centralised way, with every region relying on an independent family fund, however all IT services being provided globally by the French National Benefit Fund (CNAF).

After some delays, the CRP-GL got involved to work on an innovative approach to sort out these issues and to work out a project plan to tackle the commuter problem. To find an answer to this tricky situation, a two phases plan was defined. The first phase had to quickly realise a production ready system, able to handle the French border commuters. The proposed solution imported family allowances data from the French local benefit funds of Metz and Nancy and computed on a semestrial basis the difference between French and Luxembourgish allowances. Development started beginning 2005 and this semi-automated solution went into production in August 2005. It was extended to Belgian and German commuters in 2006. This phase, which can be considered as a preliminary work, is not being discussed in detail in this paper. The second project phase consists in developing an extendable IT system able to offer fully automated handling of the French commuters. This solution must perform the monthly computation of the family allowances and synchronise data between Luxembourg and French Family Funds. The results of this second project phase are currently in a pre-production phase at the CNPF and production is scheduled for October 2007. This modular system is supposed to be extended in order to handle all Luxembourgish commuters within a 2 years timeframe.

## **2.2 Solution description**

As mentioned above, this chapter puts the focus onto the second project phase. The proposed solution had to show operational results within 12 months, while it had to remain extendable to handle on a mid-term basis the family allowances for all neighbour countries. Another key aspect of the proposed solution was to offer extensive verification and validation mechanisms, in order to avoid incorrect or double payment of the family allowances.

The error detection is particularly tricky as the French and Luxembourgish Family Allowances Funds are involved. An extensive exchange protocol, composed of 3 sub-components, had to be specified and implemented to automatically handle those error conditions:

- The first part details how master data concerning the citizens involved in the cross-border processes are exchanged between the French and Luxembourg Family Benefits Fund. This section of the protocol specification also defines the active process for a given citizen, with eventual suspensions of the payments for a given month.



- The second part consists in a detailed error handling protocol. When abnormal situations are detected, the family funds are informed and the payments are suspended. Dedicated message exchanges have been defined to deliver status updates to the peer country in order to avoid incorrect payments. Due to the international character of these processes, it is indeed hard to get badly paid money back, especially if the involved citizens no longer live in the involved countries.
- The last part of this communication protocol handles normal/regular data exchanges between the French and Luxembourgish benefits funds. The exchanged data inform the peer country of the paid allowances and define the appropriate feedback.

During this project, a close collaboration between the French and Luxembourgish IT teams is mandatory in order to overcome organisational constraints. In addition, the communication protocol has to take the differences between the French and the Luxembourgish IT systems into account and to guarantee compatibility with both environments. Luxembourg has the advantage of being able to start with a new IT system with limited historical data and no technical constraints. The French environment is mainframe based and in production since several years. All new developments have to be carefully thought through, realised and tested. By no means new developments may negatively impact the running processes and the daily operations. The different design methodologies adopted by the two development teams also has consequences onto the working plan and the project schedule. The Luxembourg team uses agile development approaches based on the Scrum concept, whereas the French team uses the classical waterfall approach. As a consequence, the data exchange protocol has to be specified and implemented following a waterfall approach. As a consequence this sub-task has to be decoupled from the back-end system design.

This second project phase started in September 2005 and, since March 2007, is progressively moving into production stage.

Below we concentrate on the Luxembourgish part of the cross-border project, by exposing the design decision, the business processes and data manipulations at the CNPF. The core application is built around the validation workflow, which consists of several controls depending on the particular situation of the involved citizens and the corresponding allowances. The main workflow collects the appropriate data, coordinates the validation process and computes the appropriate results.

## **3 Integrating BPEL in a local application**

### **3.1 Motivation**

The first definition of the business processes are realised by means of EPC (*Event-Driven Process Chain*) ARIS diagrams in close collaboration with the CNPF business analysts. The EPC diagrams are handed over to the project team and are manually translated into BPEL workflows. We use a BPEL engine to handle workflow execution, as it is an orchestration language for web-services, initially designed by IBM and then standardised by OASIS. The language provides a way to describe the behaviour of business processes enabling transactions with remote services and ensuring interactions between them. The basic tasks of BPEL are service calls, message reception, message filtering, conditional routing and a compensation mechanism to recover from external failures. The processes are executed inside a BPEL engine, which offers management functionalities like the creation and the termination of processes according to a basic lifecycle mechanism.

Using a workflow engine to execute the underlying business processes provides an easy way to separate the behaviour of a process from the rest of the business logic. Thanks to this property, we can easily adapt a process to new requirements. Changing a test or adding a service access to a process can easily be done as the workflow is clearly separated from the rest of the code. Furthermore, most of the BPEL process editors provide a graphical representation of the BPEL process. Even if these representations are not normalized, they are informative enough to be used during discussions with non-IT people at the CNPF.

The environment of the CNPF and the nature of the processes lead our choice towards the BPEL solution. The use of BPEL is especially adapted when orchestrating fully automated workflows without human intervention, running in a distributed IT environment [Si05]. In our particular project context however, the prime goal of BPEL is to coordinate local services and to ensure execution of business processes. As a consequence several adjustments are mandatory to adapt the BPEL engine to our special needs.

### **3.2 Benefits of BPEL combined with agile development**

This section shows how, in an iterative development environment, BPEL workflows can facilitate the definition of business processes. We want to emphasize, that a BPEL based business process definition approach, is particularly adapted to small and incremental development cycles. The defined business processes have to be generic and flexible enough to follow the actual status of the underlying development of business functionalities. In practice, business processes have to be extended on a regular basis in order to integrate new business functions realised by the development teams.

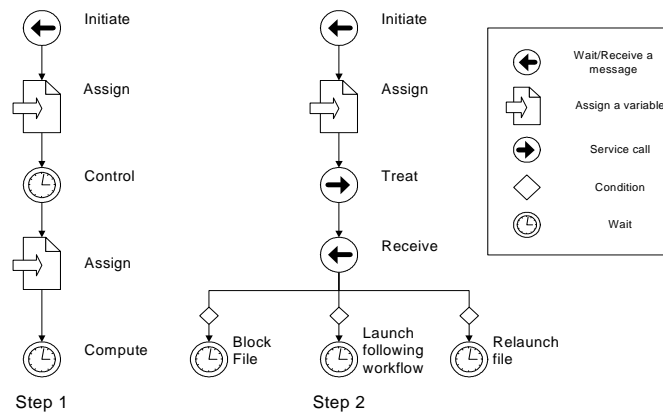


Figure 1. Example process: the first steps

The first modelling task consists in defining an ARIS EPC skeleton of the underlying business process. This initial BPEL process contains some place holders in form of wait actions, which are progressively replaced over the various production cycles. The next modelling steps consist in refining these processes. We distinguish between two kinds of refinements:

- Refinements, which integrate newly implemented tasks and business functions
- Refinements which modify the initial process by adding new structural elements in order to represent significant workflow changes

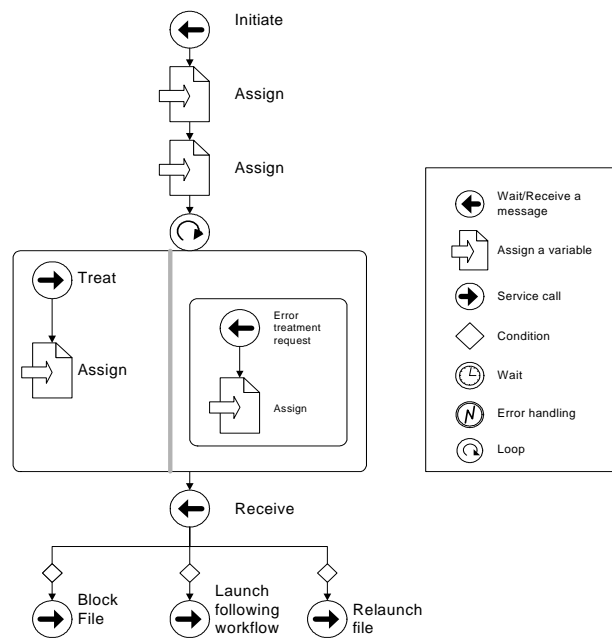
A typical example of these types of refinements of the BPEL workflow is given in Figures 1 and 2. For a better readability, we use a graphical representation of the BPEL processes. In Figure 1, we have the first version of the process, where potential external calls are replaced by wait tasks. In the second step we use two new features: the control task and a choice for the last step of the process. In the last step presented in Figure 2, we introduce a loop on the different files controlling the received data and an error handling. The left part of the “loop box” corresponds to normal behaviour; the right part corresponds to the error handling.

This example shows that BPEL processes are well adapted to iterative enhancements through short development cycles. The systematic refinement of the processes has two advantages: the process can easily be adapted to only access available services and though have testable workflows, and the multiple cycles of development give us many opportunities to correct possible errors introduced in previous modelling phases.

### 3.3 Integration issues

As the BPEL engine is used in an unusual way to orchestrate local services inside an enterprise application, we have to adapt the underlying architecture as well as the behaviour of the BPEL engine to fit these special needs. The architectural changes and adaptations are discussed in this section together with some best practices identified during the modelling process of the BPEL workflows.

The BPEL standard heavily relies on web-services. All communication with the BPEL engine relies on this technology, which introduces a certain performance overhead. Extensive performance tests however showed that this overhead is marginal compared to the underlying computations and as a consequence, the proposed solution mainly has to try to optimise the number of required web-service calls.



Step 3  
Figure 2. Example process: the final step

Using a BPEL engine at the core of the enterprise application requires the development of appropriate communication interfaces between the workflow engine and the other parts of the project. We consider three types of communication:

- Communication of Data from the application to the BPEL engine: the application server sends messages to the BPEL processes deployed as web services on the BPEL server. These messages can either start a new process or respond to an active process waiting for a specific event.

- Message flow from the BPEL engine to the application: the BPEL server accesses business functionalities deployed as web services on the application server. We use a facade pattern [Ga97] to realise the interface between the web services and the real implementation of these functionalities.
- Communication inside the BPEL engine: the main process dispatches incoming messages to the appropriate BPEL sub processes. This is done using the correlation feature offered by the BPEL language. Each sub process is accessed as a web service by the main process.

The main difficulty in this collaborative context is to ensure coherency between the processes related to the various actors in the communication. As the BPEL solution is process oriented, it is message centric. This means that the behaviour of the processes depends on the received messages and is not state-transition oriented. Consequently we have to integrate a mechanism introducing this notion of state inside the processes. This is done by means of a coordination component in charge of the synchronisation of the application state and the BPEL workflow state. The proposed coordination component can be divided into 4 parts:

1. A *State Machine* framework deployed on the application server. This framework is used by the enterprise application to trace the expected state of the BPEL process.
2. The coordination component offers query possibilities on the BPEL engine, checking if the process states inside the application code and the workflow engine are identical.
3. Automated handling of error conditions also relies on the coordination component to restore consistent status at the application and the workflow level. This error handling may result in rollback operations.
4. An event correlation module makes sure that incoming messages only influence the concerned processes. For instance, a main BPEL process catches all the incoming messages and dispatches them to the sub processes. A special identifier determines the process instance concerned by a given message. A sub process catches an incoming message only if it is actually waiting for this particular type of message. This offers an additional degree of protection and increases the reliability of the overall solution.

## 4 Model Driven Software Development and agile method

Model Driven Software Development (MDS) is a core technology of the proposed project. It helps to encapsulate most of the underlying technical aspects of the enterprise application and to focus development efforts onto the business part. The UML based platform independent model (PIM), describing the technical aspects of the project, is enhanced by several stereotypes to obtain a platform specific model (PSM). This PSM is used as input for the generator framework to produce platform specific code. In addition to the generation of the persistency layer by means of Enterprise Java Beans (EJB), several models define orchestration context behaviour. In this part, we present how MDS is used in our incremental and collaborative conception process and we give some more information about the orchestration specific models.

### 4.1 MDS development cycle

The proposed MDS approach relies on several scientific papers explaining how the agile development paradigm can be applied to MDA [Me04] or to MDS [St06]. The proposed project validates the use of the theoretical approach by a development team in a practical environment.

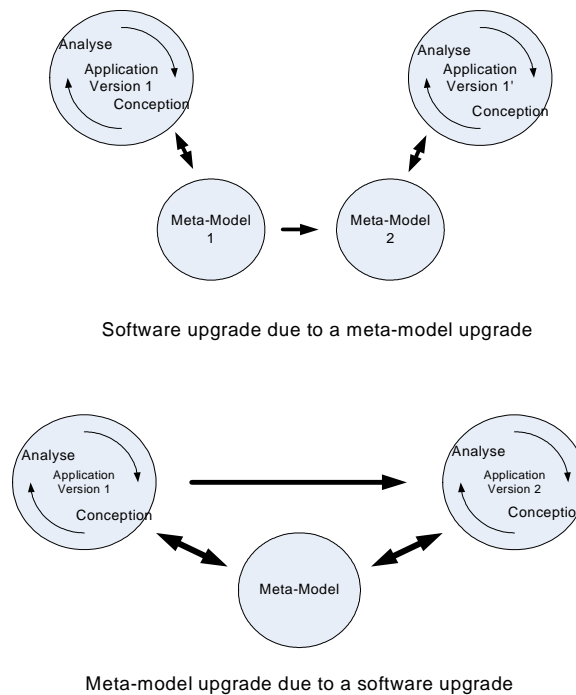


Figure 3. Correlated upgrade of the meta-model and of the application

Conceptually we distinguish the meta-model defining the general architecture and the application specific model. Both meta-model and application model evolve independently, have however mutually influencing side effects. Each development cycle relies on previous iterations, but may also require some adaptations at model, meta-model or code generation side. These changes may be caused by new functional requirements, which result in enhancements of the underlying meta-model or by architectural improvements within the meta-model. As a consequence, each development cycle can be seen as a new test for the robustness of the code and the generic aspect of the proposed models. Another advantage of the proposed approach is a quicker and more robust development. Our practical experiences are in line with the theoretical results on the common usage of MDSD technologies in an agile development environment.

Decoupling meta-model evolutions from the agile development cycles has positive side effects on the overall architecture and on the resulting enterprise applications. Figure 3 schematically shows the relationship between the application development and the meta-model evolution.

MDSD techniques significantly reduce the development effort when applied to repetitive or technical tasks, are however of little advantage when representing business logic or application specific code where manual coding is more efficient.

Another MDSD specific problem encountered during the above-mentioned project is that existing modelling tools offer only very limited multi-user support. Model sharing, or model versioning features are not ready for productive use and merging UML models is prone to error. As a workaround, we use a planning document to indicate who in the development team has ownership of the various models. This workaround introduces some overhead which is however fully acceptable in this particular project.

#### **4.2 The State Machine model**

In the overall MDSD approach we also introduce state machine support built around the state pattern proposed in [Ga97]. According to its definition, this pattern is applicable in the following context:

- The behaviour of an object depends on its state and it must change its behaviour at run-time depending on that state.
- Operations have large, multipart conditional statements that depend on the object's state. The State pattern puts each branch of the conditional structure in a separate class.

In our particular context, this definition exactly corresponds to the definition of the workflows state management. Each workflow contains two classes to handle its state transition: a *state* class that provides the core operation for state transition and a *context* class with the information that has led to the current state.

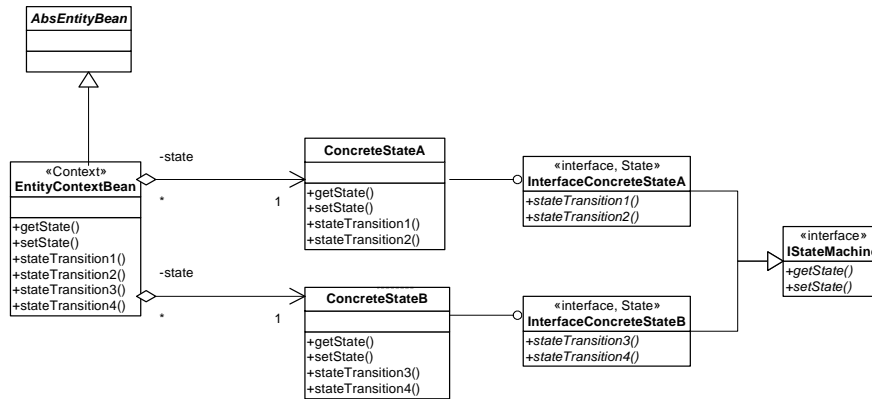


Figure 4. A State framework instance

The development team can limit itself to defining the major states (*ConcreteStateA*, *ConcreteStateB*) regrouping all states of a given state machine, as well as the appropriate transitions (*stateTransition1..4*). This UML model shown in Figure 4 is used as input and applied to the underlying meta-model. The code generator framework establishes the link with the abstract state machine, giving a generic behaviour to the workflows and with the persistence layer offering data persistence by means of Enterprise Java Beans (EJB). If the default behaviour is not appropriate, the developer may use inheritance mechanisms to overwrite the default.

## 5 Conclusion

This paper shows the benefits and difficulties encountered when integrating a BPEL engine into enterprise applications and when relying on BPEL processes to manage business workflows. The use of such a solution in an agile development process improves the flexibility of the proposed solution. It enables systematic enhancements of the business processes by adding new components to the workflow while maintaining a loose coupling between the enterprise application and the workflow engine. The resulting application shows significant better adaptability to changes. A key challenge of the proposed solution is to guarantee synchronisation between the application code and the workflow engine, by combining a message centric behaviour with a state-transition behaviour. This synchronisation component extensively uses various Model Driven Software Development techniques to offer an appropriate framework for integrating the enterprise application and the BPEL engine.



The overall experience of using a BPEL workflow engine in the particular context of Luxembourg National Benefits Fund is positive. The modernisation of the IT environment of the CNPF is ongoing while showing success stories, validating the underlying design decisions. Some additional conceptual complexity is added in the first project phases, which is compensated by a better adaptability of the overall solution. Further improvements concentrate on performance tuning in order to reduce the overhead introduced by the web-service approach of the BPEL.

## Bibliography

- [Ba06] Bauler P., Feltz F., Biri N., Pinheiro P., Implementing a Service-Oriented Architecture for Small and Medium Organisations, EMISA'06, Germany, 2006.
- [Co05] Contenti M., Mecella M., Termini A., Baldoni R., « A Distributed Architecture for Supporting e-Government Cooperative Processes », E-Government: Towards Electronic Democracy, Lecture Notes in Computer Science, vol. 3416, Springer, p. 181-192, 2005.
- [Ga97] Gamma E., Helm R., Johnson R., Vlissides J., Design Patterns – Elements of Reusable Object-Oriented Software, Addison-Wesley, 1997.
- [HF06] Hitzelberger P., Feltz F., An Interoperable Communication Platform for a Public Agency. 5th international EGOV conference, Krakow, Poland, 2006..
- [Me02] Martin R.C.: Agile Software Development: Principles, Patterns, and Practices. Prentice Hall, 2002.
- [Me04] Mellor S.J.: Agile MDA, A white paper. [http://omg.org/mda/mda\\_files/AgileMDA.pdf](http://omg.org/mda/mda_files/AgileMDA.pdf) , 2004.
- [Sc95] Schwaber K., SCRUM Development process, OOPSLA'95 Workshop on Design Patterns for Concurrent, Parallel, and Distributed Object-Oriented Systems
- [SB01] Schwaber K., Beedle M., Agile Software Development with Scrum, Prentice Hall PTR Upper Saddle River, NJ, USA, 2001.
- [Si05] Silver B., Agile To The Bone, Intelligent Enterprise, <http://www.intelligententerprise.com/showArticle.jhtml?articleID=57702677>, 2005.
- [St06] Stahl T., Völter M., Bettin J., Haase A., Helsen S., Model Driven Software Development – Technology, Engineering, Management, Wiley, 2006.



# BPMN-Q: A Language to Query Business Processes

Ahmed Awad  
Business Process Technology Group  
Hasso-Plattner-Institute, University of Potsdam, Germany  
ahmed.awad@hpi.uni-potsdam.de

**Abstract:** With the growing role business processes play in today's business life, they are being seen as an asset for the organization. With hundreds of process models developed by different process designers it would be helpful to look up the repository for models that could handle a similar situation before developing new ones. In this paper we introduce a new visual query language for business processes. The language addresses processes definitions. It extends BPMN for its abstract syntax as BPMN is a standard visual notation for modeling business processes. The overall architecture of the system in which the language can fit, in addition to the details of the query processing are also discussed.

## 1 Introduction

With the growing role business processes play in today's business life, business processes are being seen as an asset for the organization. With hundreds of process models developed by different process designers would be interested in looking up the repository for models that could handle a similar situation before inventing a new model. That is why *querying* a business process repository might be needed in such a situation. Querying of business processes can be divided into three stages, each of which has its own importance and its effect on way business is conducted:

1. Querying business process definition: this abstract model that shows the flow between activities, branching, joining, and data requirements. Querying at such level helps business analysts search for certain patterns within enterprise repository of business processes. This helps learn more about the way business is conducted, allows reusability of some business processes that might have been developed by others instead of duplication, and might help discover redundancy. Research under this category includes [VKL06, WLK06, BEKM06, LS06].
2. Querying running instances of business processes: querying here is almost a tool in the hand of administrator of a business process enactment engine to monitor the status of running processes, trace the progress of execution, make ad-hoc queries about a status of a certain process. Querying here is in flavor of Business Activity Monitoring BAM. This helps detect deadlocks, or unbalanced load on resources[MS04, BEMP07]

3. Querying execution history (logs) of completed business processes, which is also known as business process mining [vdAvDH<sup>+</sup>03] for a sample. While the purpose of business process mining is to reverse engineer the process definition from logs, purpose of querying here is wider. Depending on how much details the log gives, querying can provide us with statistics about duration of processes, bottlenecks, and the like.

In this paper we introduce a new query language that addresses the first category shown above. The rest of the paper is organized as follows, Section 2 discusses usages scenarios that are behind the design of the language in addition to requirements we extracted from these scenarios. Section 3 introduces the language informally discussing the abstract and concrete syntax of the language with an example. In Section 4 we formally describe the operation of the language. Section 5 discusses the different components of the language and how it was implemented. Details of how queries are processed are discussed in Section 6. Related work is discussed in Section 7. Section 8 concludes the paper with a view on how the work can be extended.

## **2 Usage Scenarios and Requirements**

Our work on the development of this query language was motivated with business scenarios that were extracted from a thorough scanning of literature as shown in the following subsections.

### **2.1 Change Impact Analysis**

Change in the design of process models is constant. This change could be due to new rules, obligations, or as a react to competitors improvement of service. Uncontrolled application of change would lead to degraded service or product delivered to customers rather than enhanced. This is because the change of some business activities or group of activities will lead to unforeseen affects on other business activities. Before applying change a business analyst or higher level manager should have a view on the related activities to the area of change. Impact or dependency (control flow, data flow) between activities are the major elements to be queried in this situation. Queries could take the form (for instance) like in [DC05].

### **2.2 Discovery of Frequent Process Patterns**

Modularization has been always a principle of good software design. Modularity helps localize the effect of software updates and control redundancy. The same motivation can be applied to process models as an input to a transformation that delivers an executable busi-

ness process. If certain patterns of a group of business activities appear in the same way in several models, it seems feasible to move those patterns to sub processes and replace their occurrences by calls to these sub processes. Frequent patterns have been proven to exist in real business [TLR07], one of the difficulties that faced the authors were the lack of tools to query the definition of process models to extract those frequent patterns.

### 2.3 Checking Fulfillment of Quality Constraints

Enforcement of some quality checking operations in the business process might be necessary to fulfill requirements of international quality standards like TQM, or ISO[FES05]. For instance in manufacturing processes it is necessary that the product passes through a quality check process after manufacturing and before moving it to warehouse or making it available to customer. Checking conformance of process models to this constraint can be done by querying the structure of process models.

We believe the above scenarios motivate the need of a unified access to a shared repository of business processes where structure of processes is the target of queries. The queries in such cases also share the following properties:

- Ad-hoc: usually queries are started by a claim or a doubt by a business analyst who tries to prove his claim from the underlying pool of processes.
- Iterative: usually it is not just a query that captures a snapshot and it is over, rather it is a progressive process with the nature of discovery. It begins with a simple query then the results are modified to be input for another phase of querying. The cycle stop is dependent on what the user is looking for and how far she is satisfied with the result.

### 2.4 Requirements

From the above usage scenarios we can see that the query language should target both business users 2.1,2.3 and technical users 2.2. We summarize the requirements for the query language in the following points:

1. The language should be of visual interface. The visual interface increases the usability chance for the language specially by non-technical users.
2. The language must support the navigation of process structures to answer queries.
3. Query definition goes in the same way a process definition goes. (Try to introduce the least number of new notations). This makes the learning curve for the language lower.
4. The language should support the notion of paths between nodes in the process graph.

5. The result of the query can be either the whole process model containing a match to the query or only the matching part. This should be based on a user predefined preference.
6. The ability to modify the results of queries to create new queries to support the iterative nature for the querying scenarios.

### 3 BPMN-Q

According to the requirements mentioned in section 2.4, the language we introduce is a visual language that is based on notations from BPMN [bpm06] (Requirements 1,2). The language currently addresses a subset of modeling notations available in BPMN (Activities, Events, Simple gateways). Figure 1 shows a metamodel (abstract syntax) for the language which is an extension for the BPMN Metamodel, as an extension the *Activity* metaclass describes both concrete and variable activities (will be further explained in Section 3.1), also *GateWay* metaclass is further distinguished as either a split or join. The *Connectivity* metaclass was extended with new *Path* metaclass (details in Section 4.1) to satisfy requirement 4. while Figure 2(a) shows the graphical notation for elements already in BPMN, also the notation to support the new concepts are shown in 2(b) (Requirement 3). The symbol with nested square, diamond and a star we call it a generic shape (visualization of *FlowObject* metaclass), we introduce this shape for the sake of increasing the expressive power of the language; whenever the user is not sure about the type of thing he needs in a query, the generic shape is placed. At runtime the query processor will generate and test all possibilities as will be shown in Section 6. The same idea for the diamond with S (visualization of *Split* metaclass), and that with J (visualization of *Join* metaclass) inside.

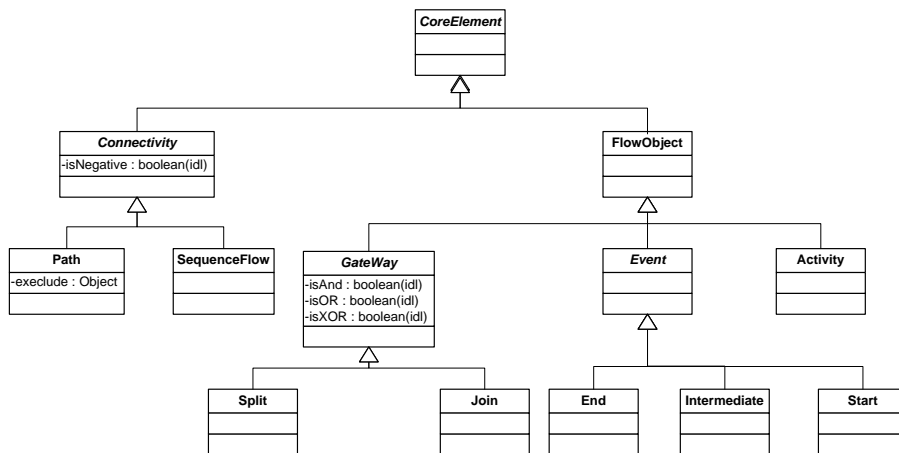


Figure 1: Language Metamodel

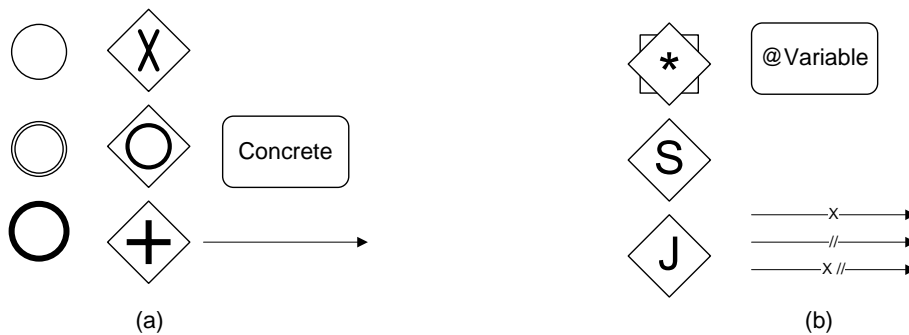


Figure 2: Language Elements

### 3.1 Example

We start with an informal introduction via an example to show how this is intended to work. Figure 3 shows a simple process model against which we will apply example queries.

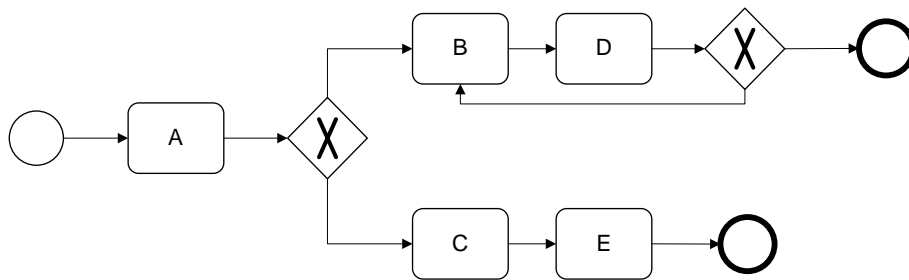


Figure 3: Sample business process model

Figure 4 contains five queries that represent simple edge expression queries (a),(b) and path expression queries (c),(d) and complex expression queries (e).

It is clear that the query representation is much like the way process models are defined (Requirement 3). There are two extensions shown in these queries:

1. Queries (a),(b) have an activity whose name is @X. This notation we call the variable activity i.e. an activity that might be bound to one or more activities in the queried process model. We prefix the activity name with the symbol "@" to inform the query processor that this is a variable activity node. In a single query expression no two variable nodes can have the same name.

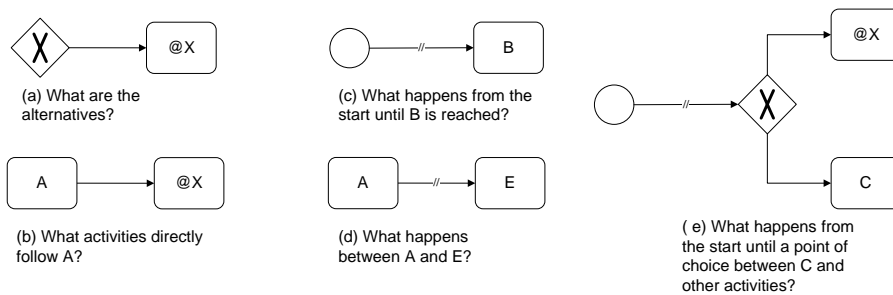


Figure 4: Sample queries

2. Queries (c), (d) the sequence flow arrow is labeled with the Symbol //. This is the symbol to represent path expressions between nodes.

To answer any query the query expression is matched to the process model. The query is answered via a set of resolution phases. If any of these phases fail, the query processing is terminated with no match. Details of processing queries are in Section 6. A single query graph maybe matched with more than one sub graph from the same process graph.

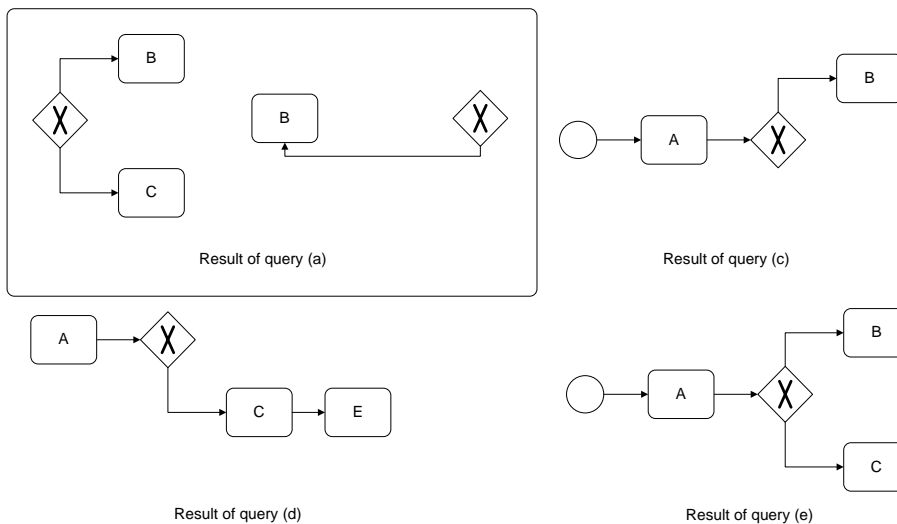


Figure 5: Queries results

Figure 5 shows the answer for the queries from Figure 4 against the process model in Figure 3. We notice that query (b) had no result because the variable node could not be bound with a concrete activity node that is a direct successor to activity A, also query (a) had two different results due to the fact that the process model had two different XOR



splits that have successor activities.

## 4 Formalization

In this section we give the formal background of both process graphs and query graphs. A process model based on BPMN can be viewed as a directed typed attributed labeled graph. Here each node in the graph has a type (task, gateway, event etc), each of them might be associated with other attributes like (name, further type details, ID) more details are in Section 5.

**Definition 4.1** A process graph is a tuple  $PG = (N, E, T, L)$  where

- $N =$  finite set of nodes.
- $E \subseteq N \times N$ .
- $T: N \rightarrow \{ACTIVITY, EVENT, GATEWAY\}$
- $L: N \rightarrow l$  is a labeling partial function .

As we have seen in Section 3 a query is also a graph. It is a directed typed attributed labeled graph, where type ACTIVITY is further described as either a concrete activity or a variable activity. Edges in query graph are also typed as either sequence flow or paths that connect two nodes.

**Definition 4.2** A query graph is a tuple  $G = (N, S, NS, P, NP, T, L)$  where

- $N =$  finite set of nodes.  $N = CA \cup VA \cup EV \cup GW$  where
  - $CA =$  set of concrete activities.
  - $VA =$  set of variable activities.
  - $EV =$  set of events.
  - $GW =$  set of gateways.
  - $CA \cap VA \cap EV \cap GW = \emptyset$
- $S \subseteq N \times N =$  sequence flow edges between nodes.
- $NS \subseteq N \times N =$  negative sequence flow edges between nodes.
- $P \subseteq N \times N =$  path edges between nodes.
- $NP \subseteq N \times N =$  negative path edges between nodes.
- $T: N \rightarrow \{CONCRETE ACTIVITY, VARIABLE ACTIVITY, EVENT, GATEWAY\}$
- $L: N \rightarrow l$  is a labeling partial function .

## 4.1 Paths

In Definition 4.2, path expression means actually all possible paths that can be found between the source and target nodes even if these paths contain cycles (due to the graph oriented nature of BPMN). Path expression is the most expensive in its computation, so when we describe how query processing takes place in Section 6, we will see how it is postponed by the query processor. The evaluation of a path expression contributes to the answer process graph according to Definition 4.3.

**Definition 4.3** A function *allpaths*:  $N \times N \times PG \rightarrow \{PG \cup \emptyset\} = PG'(N',E',T,L)$  where:

- $PG' \subseteq PG$ .
- $x \in N'$  iff:
  - $x = source$ .
  - $x = target$ .
  - $x$  lies on a path from source to target in  $pg \in PG$ .
- $\forall x,y \in N' e(x,y) \in E \rightarrow e(x,y) \in E'$

It is possible also according to the path metaclass in Figure 1 to restrict the path expression to exclude certain activity node from the path.

## 4.2 Negative Edges and Negative Paths

According to Definition 4.2, it is possible to express in the query what we call negative edges NS, and negative paths NP. These represent further Boolean conditions that can be enforced on the result of the query. For instances if two nodes A and B are connected with a negative edge in the query graph this means that in any match to the query the nodes bound to A and B must not have a sequence flow edge between them. The same applies to negative paths. Negative edges and negative paths evaluation works according to Definitions 4.4,4.5

**Definition 4.4** A function *checkNegativeEdge*:  $N \times N \times PG \rightarrow \{true,false\} = true$  iff  $e(n,m) \notin E$ .

**Definition 4.5** A function *checkNegativePath*:  $N \times N \times PG \rightarrow \{true,false\} = true$  iff  $allpaths(n,m,p) = \emptyset$ .

## 5 Architecture and Implementation

Figure 6 shows architecture for the query language. We briefly describe each component.

**Query Editor:** is a visual editor where the user can compose query in a way that conforms to Definition 4.2 and the queries look like those in fig. 4. The task of the query editor ends with passing the composed query graph to the processor component. The query editor is implemented by an extension to Microsoft Visio (see Figure 7 for a snapshot).

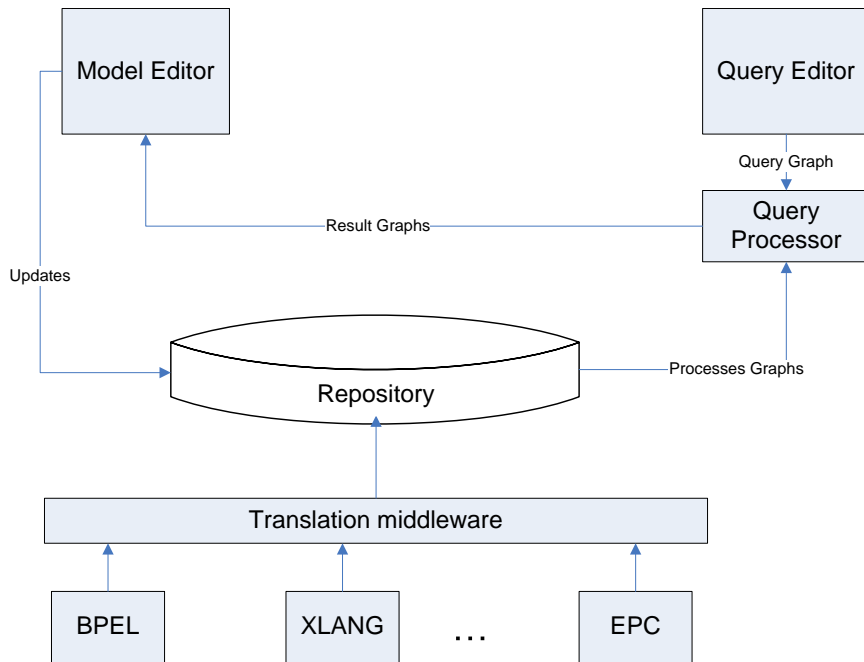


Figure 6: Suggested Architecture

**Query Processor:** receives the query graphs and works on answering it according to steps detailed in Section 6.

**Repository:** is a central database that stores an abstract uniform representation of the enterprise process models. This abstraction conforms to Definition 4.1. The database schema is simple with the following tables:

- Model(ID,Name,Description).
- Activity(ID,Name,Model).
- Event(ID,Name,Type,Model).
- GateWay(ID,Name,Type,Model).
- SequenceFlow(ID,Source,Destination,Model).
- Paths(Source,Target,Path,Model): This table is used to encode paths with all lengths that are extracted from process models. Extraction of paths comes in a post step to the storage of process models in other tables.

**Model Editor:** displays the results or messages returned by the query processor. Results can be changed by the user and then stored back in the repository, or can be reissued as new queries (Requirement 6). This is also implemented by extending Microsoft Visio.

**Translation Middleware:** translates business process definitions from specific languages syntax to the repository internal representation with metadata associated with the process graph relating it back to its source. This Middleware makes it possible to unify the query interface against different process definition languages( not currently implemented in the prototype).

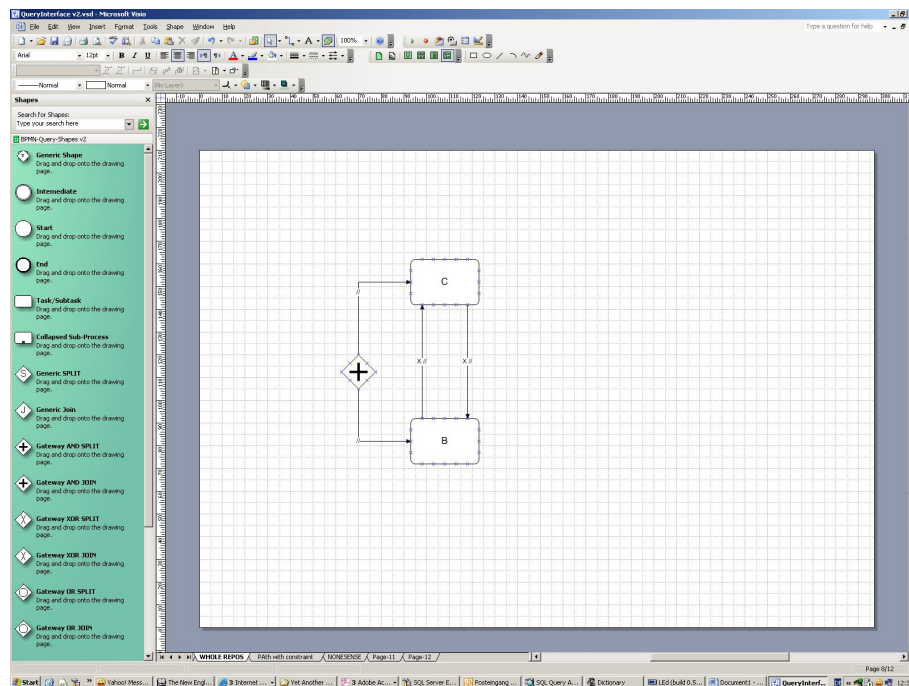


Figure 7: Query Editor

## 6 Query Processing

With the start of execution the query processor receives the corresponding query graph. To answer the query we need to search process definitions stored in the repository for those that satisfy the query graph. As we can see the repository as a graph database, one of the best ways to reduce the search space is to filter the database for only graphs which seem promising to satisfy the query [SWG02]. Within the set of graphs resulting from the filtering, we try to find an answer for the query. We have mentioned in Section 4 that all nodes are attributed by IDs. Actually we can distinguish two nodes of the same type by

ID. The process of assigning an ID to a node in the query graph with respect to a process graph is called *binding*. The query processor has to examine all possible bindings for each node. The query processor follows an approach in finding bindings that as much as possible binds a node in a way that will reduce the possible bindings for other nodes that are connected with it via *sequence flow* edges. We call this the *informed binding*. With following a set of binding steps query graph is transformed into a process graph that is the answer to the query. The query processor does the following steps for bindings 1) **Replace generic shapes**. This is a pre processing step whenever the query graph contains any of the generic shapes in Figure 2, the processor generates new versions of the original query graph in each the generic shape is replaced by shapes that can appear in process graphs. This operation simulates the principle of late binding in Object Oriented Programming OOP. 2) **Bind concrete activity nodes**. 3) **Bind event nodes**. 4) **Bind gateway nodes**. 5) **Bind variable activity nodes**. 6) **Check negative edges**. 7) **Check negative paths**. 8) **Substitute paths**.

After each binding step a new version of the query graph is created where the node is replaced by its bound node from the process graph, all edges and paths are also updated to refer to the bound node. Once a query graph version passes the first five steps, all its nodes are concrete i.e. are assigned IDs from the checked process graph. The query processor then goes to check negative edges and negative paths, steps 6, 7 according to Definitions 4.4,4.5 respectively. The last step the query processor does is to resolve path edges between nodes according to Definition 4.3 which means the maximal sub graph of the process graph in which nodes are either the two end nodes stated in the path , or a node that lies on a path from the start node to the end node. This step was postponed because (a) It is the most time consuming step, (b) We have to know exactly which nodes we look for paths between. If the query processor does not find an answer to any of the steps mentioned above, it terminates the evaluation of the query graph version. On the other hand, when a query graph version passes all the steps successfully it is now considered as an answer to the initial query and is forwarded to the Model Editor component to be displayed.

## 6.1 Performance

Table 1 shows the running time in milliseconds of queries varying in complexity, these queries are shown in Figure 8 . The queries were run against a repository containing 143 nodes, and 170 edges distributed among 11 process models. The machine used to run queries is a PC running Windows XP with 1 GB of RAM.

Query	Runtime	Query	Runtime
(a)	109	(e)	1172
(b)	219	(f)	5663
(c)	1313	(g)	3828
(d)	1141		

Table 1: Queries running time in milliseconds

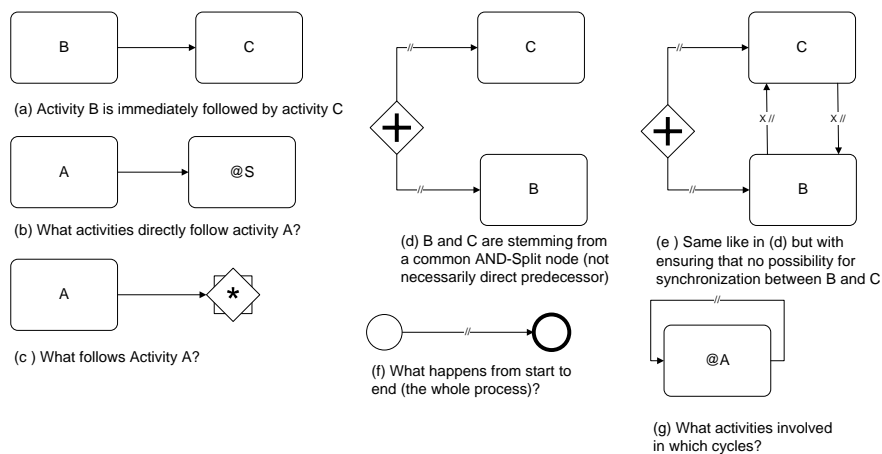


Figure 8: Queries of varying complexity

It can be simply deduced that the number of generic symbols (generic shapes, splits, and/or join), paths, variable nodes are of direct effect on the running time of a query, and this is due to that each type of them increases the number of possibilities to find a binding. Each of the alternative query graphs has to be tested for a match. In order to control the growth of generated query graphs we followed the concept of informed binding we talked about in this section. Another major effect on performance is the substitution of paths. The process of finding all possible paths between two nodes in a process graph at runtime goes in  $O(N^3)$ , of course if more than one path expression exists in a query graph the running time of the query is unacceptable. It is even worse with the iterative nature of queries, with each run all paths have to be computed again. To solve this problem we added a computation step that is executed at loading time of a new process model to repository, and each time a process model is modified. This step is to compute paths with all lengths between nodes (if there is a path). So the cost of computing a path is now constant at query execution time rather than cubic.

## 7 Related Work

In [VKL06] a repository for process definitions based on BPEL [ACD<sup>+</sup>03] was built. BPEL files are annotated with organization specific metadata, the repository was queried through a set of APIs that address at first place queries issued by programmers to select a specific BPEL file for execution based on metadata search criteria. Our work is distinguished in that we address humans at analyst level, query is based on the structural properties of process models. Similar work in [WLK06] queries the content of business process based on a framework for describing this content. The framework consists of four concentric levels: high level flow, business flow, activity, and task. Each level is associated

with metadata that are used for querying. The query goes from the broader scope narrowing the result set with each step from a level to the next level based on values specified for associated metadata at each level. Work in [BEKM06], [LS06] are considered close to our work from the point that they query process definition from structural point of view. The Business Process Query Language BPQL in [BEKM06] works on an abstract representation of BPEL files. We can distinguish our work with the handling of cycles in the process graph, a point that is not available in BPEL. Another point is the ability to query with generic expressions like generic shapes, splits, and/or joins. Querying process variants in [LS06] utilizes graph reduction techniques to find a match to the query graph in the process graph, comparing it to our work we have more concepts like the variable activities, path finding, and generic shapes which increase the expressive power of the language in addition to handling process graphs with cycles while [LS06] works on acyclic graphs only.

## 8 Conclusion and FutureWork

In this paper we introduce a new visual query language for searching repositories of business processes. The language was an extension of BPMN in its abstract syntax. Also we discussed an architecture where the language fits, the details of query processing were discussed. We have shown throughout Section 3 and Section 5 how requirements 1,2,3,4, and 6 were satisfied. Regarding requirement 5 we chose in our prototype implementation to show the matching model with highlighting the answer within it. As Future work, covering data flow, resources, and messaging between interacting processes are open areas to be included in the query language.

## References

- [ACD<sup>+</sup>03] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. Business Process Execution Language for Web Services version 1.1. Technical report, OASIS, 2003.
- [BEKM06] Catriel Beeri, Anat Eyal, Simon Kamenkovich, and Tova Milo. Querying business processes. In *VLDB'2006: Proceedings of the 32nd international conference on Very large data bases*, pages 343–354. VLDB Endowment, 2006.
- [BEMP07] Catriel Beeri, Anat Eyal, Tova Milo, and Alon Pilberg. Query-based monitoring of BPEL business processes. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1122–1124, New York, NY, USA, 2007. ACM Press.
- [bpm06] Business Process Modeling Notation (BPMN) Specification, Final Adopted Specification. Technical report, OMG, 2006.

- [DC05] Weizhen Dai and H. Dominic Covvey. Query-Based Approach to Workflow Process Dependency Analysis. Technical Report 01, School of Computer Science and the Waterloo Institute for Health Informatics Research, Waterloo, Ontario, Canada, 2005.
- [FES05] Alexander Förster, Gregor Engels, and Tim Schattkowsky. Activity Diagram Patterns for Modeling Quality Constraints in Business Processes. In Lionel C. Briand and Clay Williams, editors, *MoDELS*, volume 3713 of *Lecture Notes in Computer Science*, pages 2–16. Springer, 2005.
- [LS06] Ruopeng Lu and Shazia Wasim Sadiq. Managing Process Variants as an Information Resource. In Shahram Dustdar, José Luiz Fiadeiro, and Amit P. Sheth, editors, *Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, pages 426–431. Springer, 2006.
- [MS04] Mariusz Momotko and Kazimierz Subieta. Business Process Query Language a Way to Make Workflow Processes More Flexible. In *ADBIS'2004: Proceedings of the 8th East-European Conference on Advances in Databases and Information Systems*, pages 306–321. Springer Berlin / Heidelberg, 2004.
- [SWG02] Dennis Shasha, Jason Tsong-Li Wang, and Rosalba Giugno. Algorithmics and Applications of Tree and Graph Searching. In *Symposium on Principles of Database Systems*, pages 39–52, 2002.
- [TLR07] Lucinea Heloisa Thom, Cirano Lochpe, and Manfred Reichert. Workflow Patterns for Business Process Modeling. In Barbara Pernici and John Atle Gulla, editors, *CAiSE*, volume 4495 of *Lecture Notes in Computer Science*, pages 349–357. Springer, 2007.
- [vdAvDH<sup>+</sup>03] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters. Workflow mining: a survey of issues and approaches. *Data Knowl. Eng.*, 47(2):237–267, 2003.
- [VKL06] Jussi Vanhatalo, Jana Koehler, and Frank Leymann. Repository for Business Processes and Arbitrary Associated Metadata. In *Demo Session of the 4th International Conference on Business Process Management*, pages 25–31, Vienna, Austria, 2006.
- [WLK06] Avi Wasser, Maya Lincoln, and Reuven Karni. ProcessGene Query- a Tool for Querying the Content Layer of Business Process Models. In *Demo Session of the 4th International Conference on Business Process Management*, pages 1–8, Vienna, Austria, 2006.



# OBSE – an approach to Ontology-based Software Engineering in the practice

Andrej Bachmann, Wolfgang Hesse, Aaron Russ (University Marburg)  
Christian Kop, Heinrich C. Mayr, Jürgen Vöhringer (University Klagenfurt)

Philipps-University Marburg  
Hans-Meerwein-Strasse  
35032 Marburg  
{rodionov,hesse,russa}@mathematik.uni-marburg.de

University of Klagenfurt  
Universitätsstraße 65 - 67  
9020 Klagenfurt  
{chris,heinrich,juergen}@ifit.uni-klu.ac.at

**Abstract:** In this article we present a new approach to Ontology-based Software Engineering (OBSE) meant for practical use in enterprises and industrial projects. Following this approach, Software projects are no longer driven only by requirements and models but also by one or several ontology/ies covering their application domain. Our main thesis says that OBSE can offer similar opportunities and benefits for re-engineering and re-use in the early phases of software development as object orientation does for the later ones. OBSE is to be supported by tools which integrate ontologies in the SE process. A prototype of such a tool – based on the KCPM and EOS methodologies – is presently being developed in a joint project of our groups.

## 1 Introduction

Ontologies have been introduced as a key concept in informatics in the last decade of the previous century when the rapid growth of the internet and its services created new demands on automated agents and similar devices which require facilitated and encompassing access to domain knowledge of various application domains. Gruber has defined *ontology* as an *explicit specification of a conceptualization* [Gru 95]. Applications of ontologies in Informatics comprise the fields of Artificial Intelligence, Agent systems, Database & Information systems, Web Technology and other fields.

In the *Software Engineering (SE)* field conceptualisation has played a major role for long time, e.g. during the *analysis, modelling and design* phases of software development, where all relevant entities of the application domain with their features and relationships have to be conceptualized. This way, considerable portions of domain knowledge are elaborated in almost every application-oriented software project. Similarly, large portions of technical and implementation-oriented knowledge are worked out during the *detailed design, implementation, test & integration* phases. However, whereas *object orientation (OO)* technology has led to major achievements in re-using the latter kind of knowledge this is still a desire and challenge as far as the domain knowledge relevant for the early phases is concerned.

*Ontologies* seem to be an appropriate concept for describing portions of domain knowledge which is to be re-used across several software projects. Thus we aim for a new approach for integrating ontologies in the SE process. Following this *Ontology-*

based Software Engineering (OBSE) approach, software projects are no longer driven only by requirements and models but also by one or several ontology/ies covering their application domain (cf. fig 1).

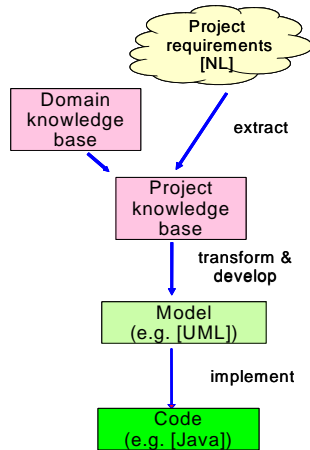


Fig. 1: An ontology-based software processing

For dealing with both questions, previous work of the two groups authoring this article can usefully be employed:

- The *Klagenfurt Conceptual Predesign (KCP)* method offers *glossaries* as an appropriate and useful instrument to deal with ontologies in the early phases of software development.
- The (Marburg-based) method for *Evolutionary, Object oriented Software Development (EOS)* offers a Software Process model which is flexible and wide enough to cover ontology-based processes as well and allows for a unique treatment of software and ontology engineering processes.

In our view, the OBSE process should be a combination of both (Software and Ontology Engineering) life cycles following some sort of *rendezvous* principle: Software Engineering projects inherit from existing ontologies in the early (analysis and modelling) phases and offer (parts of) their results for further ontology development and evolution. The first step allows re-use of domain knowledge whereas the later promotes re-use of project specific knowledge. The EOS model serves as a joint framework for defining and supporting the OBSE approach.

Furthermore we argue that the appropriate form for including domain knowledge from ontologies in the SE process is the *glossary form* making the KCP method and tools a key basic technology for OBSE. Since glossaries are modular, flexible and easy to handle for users, domain experts and engineers, they seem to be a most appropriate form for documenting reusable knowledge.

In the subsequent sections, we will first briefly compare Software and Ontology Engineering processes and then define our OBSE process model based on the KCP and

Our main hypothesis is that by combination of Knowledge and Software Engineering techniques OBSE can offer similar opportunities and benefits for re-engineering and re-use in the early phases of software development as object orientation does for the later ones. Of course, such an approach has to be supported by tools for software engineers who want to integrate ontologies in their SE process.

Two major questions arise when we investigate approaches to OBSE in more detail:

- (1) What is the appropriate form (and language) for expressing ontologies in the SE context? **and**
- (2) How can the SE process be extended to an OBSE process including the use and evolution of ontologies?

EOS approaches. In the last sections we will deal with tool support for OBSE and include a rough sketch of the tool prototype under construction.

## 2 Software and Ontology Engineering process: brief comparison

*Ontological Engineering* has been advocated by Mizoguchi [Miz 98] and analogies (as well as differences) with Software Engineering and their processes have already been discussed, e.g. in [G-L 02] and [Hes 05]. In the following table, both fields are compared and of some of their outstanding characteristics and properties are listed (cf. Fig. 2).

	<b>Software Engineering</b>	<b>Ontology Engineering</b>
<b>Target groups</b>	<ul style="list-style-type: none"> <li>• Software managers, engineers and users engaged in a particular project</li> </ul>	<ul style="list-style-type: none"> <li>• Domain experts, ontology builders and users dealing with many projects in a particular domain</li> </ul>
<b>Principal requirements</b>	<ul style="list-style-type: none"> <li>• Functional requirements regarding system procedures, correct output etc.</li> <li>• Quality requirements re. user friendliness, reliability, performance ..</li> </ul>	<ul style="list-style-type: none"> <li>• Trustability and consistency</li> <li>• Compatibility and accessibility from many projects and applications</li> </ul>
<b>Project duration</b>	<ul style="list-style-type: none"> <li>• determined, limited for one project</li> </ul>	<ul style="list-style-type: none"> <li>• undetermined, unlimited</li> </ul>
<b>Process structure</b>	<ul style="list-style-type: none"> <li>• often sequential: phases, activities, but also iterations and cycles</li> <li>• Sub-processes for components or increments</li> </ul>	<ul style="list-style-type: none"> <li>• mostly cyclic, cycles maybe grouped in phases</li> <li>• Sub-processes for developing or revising subdomains</li> </ul>
<b>Process models</b>	<ul style="list-style-type: none"> <li>• Waterfall, incremental, component-based, prototyping, spiral-like</li> </ul>	<ul style="list-style-type: none"> <li>• incremental, component-based, evolutionary</li> </ul>
<b>Concepts, languages and tools</b>	<ul style="list-style-type: none"> <li>• Use cases, natural language, modelling &amp; programming languages (e.g. UML), diagrams, pseudo code</li> <li>• Tools: Editors, modelling tools, compilers</li> </ul>	<ul style="list-style-type: none"> <li>• Natural language, glossaries, tables, semantic networks, topic maps, conceptual graphs, ontology languages</li> <li>• Tools: Ontology editors, modelling tools</li> </ul>
<b>Results and products</b>	<ul style="list-style-type: none"> <li>• project-specific, (relatively) short-term oriented, usable for particular application</li> </ul>	<ul style="list-style-type: none"> <li>• spanning many projects, long-term oriented, re-usable, "sharable" among many organisations and projects</li> </ul>

Fig. 2: Some characteristics of Software and Ontology Engineering

As the table shows, ontology development resembles software development in various respects but there are also significant discrepancies between the two kinds of processes, e.g. resulting from different target groups, contexts and requirements. For a more comprehensive discussion we refer to [Hes 05].

### 3 KCPM: A glossary-based approach to Software and Ontology Engineering

In order to motivate the *KCP method (KCPM)* as a missing link between software requirements and software modelling/design we will briefly present our glossary approach, the concepts and representation forms of KCPM. Afterwards the appropriateness of KCPM will be discussed.

KCPM was introduced to support the requirements elicitation process. As described in the previous section there are a lot of similarities between software engineering and ontology engineering. During the first phases of *requirements engineering* and *knowledge acquisition* the developers (ontology builders and software engineers, resp.) have to communicate with experts<sup>1</sup>. Performing this task the engineer is more like a doctor who has to ask the right questions or like a pilot who has to check that everything is working before he starts the engine. The paradigm of such a check list which supports the task to formulate the right questions directly leads to the idea of using glossaries as a concept for representing requirements. In the KCP methodology, a glossary is employed as the central knowledge base for gathering, storing and communicating domain knowledge during the requirements capture and modelling phases of software projects [M-K02].

In particular glossaries have the following advantages:

- Domain experts are mostly familiar with glossaries since they use them in their daily work.
- It is easier to find a gap in a glossary-like specification (the regarding column is empty). Thus a glossary is like a check list.
- Information that belongs together (e.g. regarding the same concept) is associated with that concept. Thus the information is collected in a very compact manner.
- The structure of a glossary is standardised and predefined.
- The semantics of the key terms of glossaries (e.g. the column names) are predefined.
- The glossary type (e.g. thing type glossary, operation type glossary) as well as the several columns within these glossary types provides a first classification of the collected information.

#### 3.1 Small set of modelling concepts

The glossary is built up by few kinds of (table-like) type descriptions the most important of which are: *thing types* and *connection types*. In order to support the glossary building task, linguistic techniques such as natural language text analysis are employed and supported by corresponding tools [FKM+ 00]. Glossaries may be transformed into conceptual models or UML-like class structure diagrams according to a set of laws and transformation rules in a *semi-automated* way.

---

<sup>1</sup> We assume that every person is an expert on a certain domain. In the most specific way he/she is the expert of the tasks he/she has to do in an enterprise.

**Thing-type** is a generalisation of the UML concepts class and attribute. Thus, typical thing types are e.g. *author*, *book*, *contract* as well as descriptive characteristics like *customer name*, *product number*, *product description*. It seems to be easy to decide, which of the above examples is a class and which one is an attribute, but what about a concept used in a domain which is not well known by the designer (e.g. the concept *ICD10* in the medical domain). Following KCPM the question whether the concept is a class or an attribute is not a primary question but this will be decided later and be supported during the mapping process. Instead the system analyst can concentrate on gathering additional information for that concept, which is much more important during requirements analysis. Meta-attributes which head the glossary columns (e.g. *Examples*, *Synonyms*, *QuantityDescription*) give hints to ask the right questions.

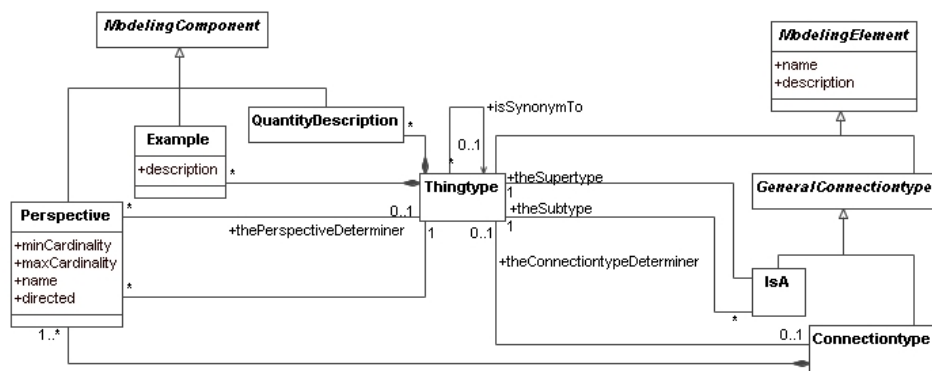


Fig. 3: Overview of the KCP meta model

Things are related within the real world. To capture this, the KCPM model introduces the concept of **connection-type**. Two or more thing-types can be involved in a connection-type. This is based on the NIAM (ORM) object/role model [N-H 89]. A sentence (business rule) leading to a connection type could be the following: *Authors write books*. The model is open for specific semantic connection-types (possession, composition, generalization, identification etc.) e.g. *An ISBN number identifies a book*.

This glossary approach works similar for all the other KCPM concepts (connection type, operation type etc.) which are described in other papers. In the meta schema concepts and columns (meta attributes) are distinguished in the following way. KCPM concepts are derived from the class *ModelingElement* and columns from *ModelingComponent*.

### 3.2 KCPM as a link between domain ontologies and SE

As was mentioned before, KCPM was introduced as a requirements modelling language. However looking at the modelling concepts and the representation concepts of KCPM, KCPM can be also seen as a link between Ontologies and Software engineering. In order to motivate this assumption it has to be shown that

- (1) There is a general relationship between KCPM and ontologies
- (2) KCPM can be used for conceptual models in the software engineering domain.

To justify the first statement we need to answer first the following questions: What is the purpose of an ontology? What are possible ontology representations?

Since Gruber's article [Gru 95] ontology is understood as a means for knowledge sharing. In [Gua 98] domain and task ontologies describe the vocabulary related to a generic domain. This distinguishes domain ontology from a conceptual model where the vocabulary has to be refined for the project specific purpose.

For the second question we have to take a look at the several representations of ontologies. These representations range from lexicons, notion lists, and topological maps to formal specifications. Most often glossaries are used to describe the notions.

Comparing this with the KCPM approach we can conclude: KCPM has a representation concept that fits very well into possible representation concepts of ontologies. Furthermore, in the previous section the advantages of such a representation were already stated. These advantages can also be applied to ontologies.

As static concepts KCPM offers just thing types and connection types. From our experience in many modelling projects, we learned that the distinction between classes and attributes is often artificial, premature or project dependent. A notion which might be modelled as a class in one project (e.g. *address*, *driver*) might be modelled also as an attribute in another project. If we abstract from this distinction using thing types for both in common, then this fits much better to the ontology level where the involved parties have to concentrate on getting a *shared knowledge and understanding* of a domain.

To motivate the second statement from above, we refer the reader to [M-K 02] where we have described in detail how thing types and connection types can be mapped to conceptual models used in the SE domain (i.e. UML diagrams). Furthermore it was explicated in [V-M 05] that integration on the conceptual level (e.g. using KCPM) has advantages compared to integration in the later phases of software development. This has also a beneficial impact on the process of shared knowledge generation. If two or more involved parties want to adjust their knowledge to a common shared knowledge it is much better to abstract from terms like class and attributes.

### 3.4 General Overview of the OBSE cycle

Based on these connections between KCP glossaries, UML, and ontologies, a unification and combination of the ontology development life cycle for a certain domain on the one hand side and software project life cycles concerning the same domain on the other is promising. A first glance on the combined life cycles is shown in fig. 4. The key for this unification is the use of KCP glossaries in both life cycles: On the ontology side, the domain knowledge is captured in a glossary before it is (partly automatically, see above) transformed to UML or some other ontology language (OL) representation. On the software project side, project specific domain knowledge is extracted from the requirements after elicitation and stored in a glossary-like knowledge base which is transformed to UML models in the described way and then further to code of some *programming language (PL)*.

In our unified process, KCP glossaries are used in order to support the requirements and ontology engineering processes as well as the exchange between both of them. Since in this phase requirements are mainly collected through user interviews and the study of *natural language (NL)* documents, ambiguities can easily occur and lead to communication problems. On the other hand, domain ontologies usually contain data that is reviewed by domain experts and that describe the important concepts and relationships of the specific domain.

These data can be used in various ways to replace or to complement information gathered in the "normal" requirements engineering process. For example, information that was collected through usual requirements engineering techniques might be verified while matching it against the ontology data. This way, discrepancies between the ontology description and the data from other sources might be identified. These conflicts are often caused by ambiguities or imprecision in the requirements documents and must be resolved. Moreover, the ontology data might complement the previously gathered requirements. This can be accomplished combining the domain glossary and the project glossary by *schema integration* (cf. chap. 3.3).

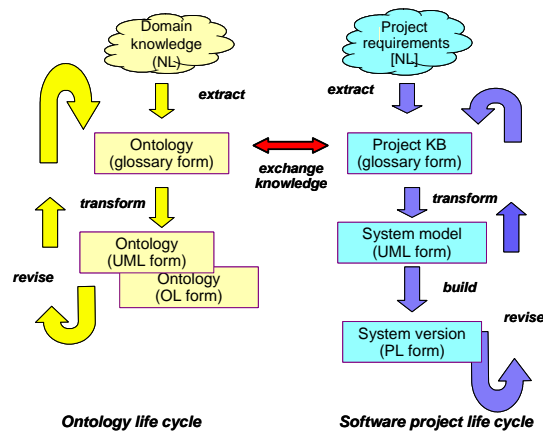


Fig. 4: Ontology and software project life cycles combined in a rendezvous manner

This combined and integrated ontology can be modified in order to meet project specific needs. Using this ontology the project runs through the following design and implementation phases of the software development life cycle. During the final project phases domain knowledge that has been gained during the project and which was not yet part of the domain ontology or which should be used for its revision can be exported and then used within an integration process modifying the original domain ontology.

## 4 The EOS model and its use for a combined OBSE process

### 4.1 Basic concepts of the EOS model

In order to obtain a uniform view on both Software and Ontology Engineering processes as sketched above, the EOS model can successfully be used (cf. [Hes 03], [Hes 05]). It has been developed in order to support *Evolutionary Object oriented Software Develop-*

ment and it offers a high degree of flexibility and scalability for both software managers and engineers when dealing with complex projects which include many components and highly concurrent development processes. Its use for Software Engineering projects has been described in detail elsewhere (cf. e.g. [Hes 96], [Hes 97], [Hes 03]). Among its key concepts are:

- *Component-based structure and architecture-driven, uniformly structured development cycles:* Unlike most of its traditional predecessors, the EOS model binds development cycles to the building blocks of the system architecture. All development cycles consist of the four main activities analysis, design, implementation and operational use – irrespective of its occurrence at the system, component or module/class layer (cf. right part of fig. 5). This way, development processes become highly scalable and flexible.
- *Multiple, mostly concurrent development cycles and evolutionary software development:* Re-development or revision cycles may be activated and performed on demand at any architectural level and thus system evolution is encouraged and supported by the EOS model. Concurrent development cycles are synchronised by means of revision points, i.a. predefined points in time where certain activities have to be finished and their results are available for review or delivery.

#### 4.2 Ontology development described in EOS terms

Ontologies are preferably structured as hierarchies (cf. [Gua 98]), e.g. consisting of the three levels:

- universal ontology
- discipline ontologies
- domain ontologies

Further decomposition may lead to smaller units – let us call them *ontology components (OC's)* in analogy to the EOS terminology. Ontologies are in a continuous process of *evolution* – this leads to the requirement of independent, often concurrent OC development cycles. These can be well defined using the general EOS schema for describing development cycles:

- *Ontological analysis* aims at defining the OC and delimiting its boundaries, identifying potential applications, analysing the relevant terminology, building a taxonomy, describing terms as glossary entries, and dissolving terminological conflicts. The analysis results in a first version of the OC glossary.
- *Ontology design* deals with defining a (sub-) structure of the OC, defining facts and rules, building UML maps of the OC, check-ups and comparisons with other glossaries, modifying and (re-) structuring the taxonomy and particular glossary entries, dissolving terminological conflicts.
- *Ontology implementation and integration* aims at translating the OC and its elements into a formal ontology language, checking syntax and semantics, integrating and validating sub-ontologies, comparing and unifying conflicting terms, dissolving terminological conflicts



- *Ontology operational use* comprises publishing the OC, checking, validating its ingredients and asking for and receiving feedback, adapting the OC to super-/neighbour ontologies, looking for requests for revision, initiating revision (if necessary).

If we consider an ontology as a hierarchy of OC's, we can use the EOS model as a generalised life cycle model for developing ontologies of any size in an evolutionary way: Ontology development is then a complex process of concurrent OC developments. One particular OC may be developed in two ways: either (in a *top down* fashion) as part of the (already existing) encompassing domain ontology, or (*bottom up*) as part of a software project located in the concerned domain. In the latter case, the (ontology-related) results of the project have to be integrated into the encompassing domain ontology.

Of course, this step is not easy and is quite similar to the well-known schema integration problem. It consists of combining the separate ontology parts to a single one by identifying communalities and conflicts, while resolving the latter. However, the continuous update of domain ontologies through project ontologies allows the knowledge bases to be kept up-to-date and always relevant for further projects. Such an approach can only be successful if the domain ontologies are of high quality and if sophisticated and well-proven comparison and integration techniques are used.

### 4.3 Outline of an EOS-based OBSE process

With the above prerequisites, we can define an *OBSE process* as a combination of project and ontology development cycles. We consider a software project concerning an application domain  $D$  for which a domain ontology  $O_D$  already exists. The subset of  $O_D$  which contains all definitions and explanations relevant for our project  $P$  is defined as an ontology component  $OC_P$ . An OC development cycle may be attached to  $OC_P$  as outlined above and depicted in the left part of fig. 5.

Two so called *bridges* support the exchange of information between the domain ontology and the software project life cycles (shown on the left and right part of fig. 5, resp.). The first bridge (labelled by "*import*") is relevant in the analysis phase of the software development process. If the resulting system enters the phase of operational use and has proven stable enough, the second bridge to the ontology life cycle (called "*export*" bridge) becomes relevant.

In particular, ontology analysis of  $OC_P$  results in a glossary  $GL_P$  which can be transferred to the software project  $P$  via the *import* bridge. According to the EOS guidelines, system analysis for the project  $P$  starts from requirements which delimit the scope of the system  $S$  to be built and of its application domain  $D$ . Moreover, system analysis steps are now supported by the imported definitions of  $OC_P$  (cf. fig. 5). This way, project  $P$  profits from previous ontology work on the domain  $D$  as aimed by the overall OBSE approach.

Ontology import may also be broken down to the component structure of  $S$ . According to the EOS model, the system analysis and design steps for  $S$  lead to a component structure consisting of components  $X_i$  ( $i = 1, \dots, n$ ). Let us suppose  $X_1$  to be the component responsible for the application domain  $D$ . Then the analysis of  $X_1$  implies importing the definitions of  $OC_P$  via the import bridge. If there are more components

relying on the domain  $D$  and its definitions, the same *import* procedure applies for all these components.

Following the analysis steps, the software project  $P$  goes on as prescribed by the EOS model: Components are designed, may be decomposed into sub-components and modules which run through their own development cycles. Implemented modules are tested and integrated to subsystems which in turn are integrated (in an incremental or whatever way) to form the envisaged system  $S$ .

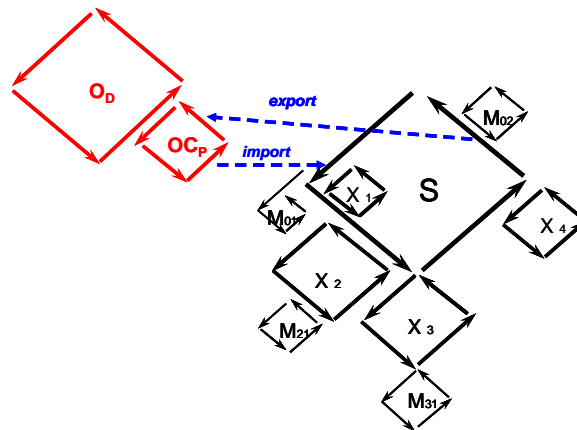


Fig. 5: System development and ontology life cycles interconnected

*Operational use* is the last step of every EOS development cycle – and, in particular, of the overall system development cycle (marked by "S" in fig. 5). This step is the second anchor point for OBSE-related actions: A review of the project and its results implies a particular resume of its contributions to the domain ontology. If there are any significant enhancements or modifications, these are transferred to the ontology development process of  $O_{C_P}$  via the *export* bridge. Again, these contributions may be located in some component(s) of the system  $S$ , viz. in their implementations and are to be extracted via the project glossary.

## 5 The OBSE tool and prototype

The *OBSE tool* is intended to support software engineers who want to work along the OBSE process. The main purpose of the tool is to combine the KCPM based ontology development with the EOS software developing process. In the centre of this integration are import and export bridges (cf. fig. 5 and process description above).

- For import activities, the tool provides support by transferring elements of the domain ontology into a project knowledge base during the *analysis* phase.
- On the export side, information gained from a project is transferred to its respective domain ontology via the second bridge offered by the tool. Typically this process takes place in the *operational use* phase.

These bridges work on the glossary level. This means that the elements of the domain ontology are transferred via import functions into the KCPM glossary of a project and vice versa by export functions. However, often the knowledge to be transferred is not given in glossary form but maybe, e.g. in UML form. In order to support the transfer in these cases as well, transformations from KCPM glossaries to UML and back have been implemented [SMK 04], [Rus 07]. These transformations ensure an indirect export of UML models typically developed in software projects as well as the use of imported glossary elements in projects working with UML.

In the majority of cases both import and export requires an integration of KCPM glossary entries into existing KCPM glossaries. For example, at the beginning of a project a lot of information about the associated domain was extracted from project requirements into the project knowledge base (i.e. a conceptual model in glossary form) [M-K 02]. This model can be enhanced by elements from the domain ontology using import functionality of the OBSE tool. This is an integration process which requires specific merge functions for glossaries (cf. [V-M 05]). The integration steps must be seen as semi-automatic. Meaning the tool user can choose which elements are transferred, and specify the rules that are to be used during each integration step. The export of glossary entries into the existing domain ontology is done analogously. Since both integration functions work on the glossary level, they have been implemented in a uniform manner in the OBSE tool.

Besides the import-export functions which are essential to the OBSE process, the OBSE tool offers other features which support the management of glossaries on the domain ontology side as well as on a project knowledge base. This is not limited to graphic or table-like views of data with integrated edit function but also incorporates a built-in and always adjustable OBSE project description. This offers the user a help facility e.g. defining roles, activities and artifacts of the process, guides him/her with iteration and activity descriptions and combines the use of the tool with planning and designing activities of the process. By integrating the OBSE process description into the OBSE tool we hope to promote the ability to learn and consistently use both.

One fundamental question concerns the architecture and platform of the OBSE tool: Which architecture would best be suited for the tool having above mentioned goals in mind? For various reasons (detailed in the following) we have decided for a *PlugIn based architecture*, rooted in the *Rich Client Platform (RCP)* – at least for our first OBSE tool prototype.

RCP is a framework consisting of a relatively small core which is extendable for specific functionality via PlugIns and compatible with many operating systems.

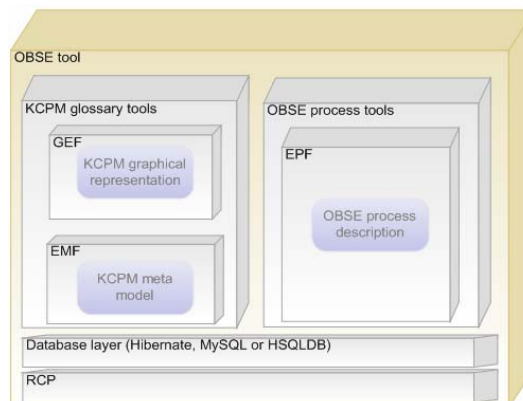


Fig. 6: OBSE tool structure

A well known implementation of RCP forms the basis of the Eclipse toolset [L-M 05]. This will be used as a platform of our prototype and allows us to construct the OBSE tools as a collection of multiple PlugIns.

Eclipse RCP elements such as perspectives and views permit the definition of different views on data for different roles and tasks in the process while maintaining uniform usage and surface. This way, different PlugIns appear to the user as a almost monolithic, homogeneous system. The *Eclipse Process Framework (EPF)* plays a key role in the OBSE tool development and the implementation is eased by its RCP based structure. It allows a description (with roles, activities, artifacts etc.) of the OBSE process to be generated and published via the tool. Another advantage of the framework is the ability to adapt process descriptions to one's own needs. Should it be necessary to define additional tasks or replace artifacts with its own variants in a project that is being carried out with OBSE this can be achieved with the help of the EPF underlying process part of the tool. This supports the scalability of the OBSE process, i.e. it makes it usable for small as well as for large projects.

We see additional advantages in other frameworks from the Eclipse Foundation. This includes the *Eclipse Modeling Framework (EMF)* and *Eclipse Graphical Framework (GEF)*. The KCPM meta model is defined with EMF. The classes of the meta models used by other PlugIns are generated by this model, including interfaces and facade classes. This provides the consistency of the KCPM meta model and the code which belongs to it. GMF is a powerful tool for implementing the graphical representations of glossary entries (cf. fig. 6). The OBSE-Tool prototype, which is currently being developed implements the above mentioned concepts and will presumably be finished by end of 2007.

## 6 Outlook: OBSE and MDA

*Model Driven Architecture (MDA)* [OMG 03] is a model centred approach which is expected to play an growing role in future SE. In the terminology of MDA three different types of models are defined: *Computation Independent Model (CIM)*, *Platform Independent Model (PIM)* and *Platform Specific Model (PSM)*. The idea is to engineer an abstract model which then can be used to generate more specific models for different target platforms. These models can be described in a modelling or natural language – note that PIM and PSM are usually expressed in UML. MDA concentrates on PIM and PSM and their transformation. For CIM only an imprecise description can be found.

We argue that project specific KCP models are suitable as CIM (cf. fig. 7). In [GDD 06] it is pointed out that CIM can be seen as some kind of ontology and as mentioned in chapter 3, the KCP method presents an appropriate way for expressing ontologies. Beyond this similarity, project specific KCP models are created from project requirements as well as from a domain ontology whereas CIMs are expected to describe the requirements for a system and the system's immediate environment.

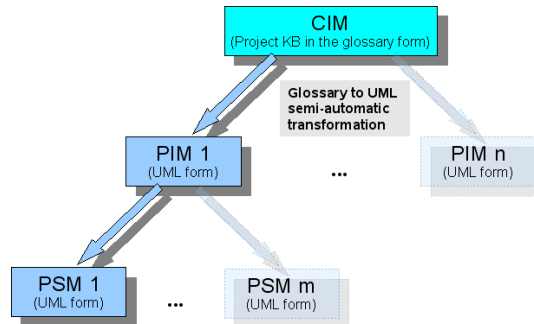


Fig. 7: OBSE and Model-driven Development (MDD)

The use of KCP models as CIM paves the way for a possible *MDA extension* going beyond the so far existing transformations which are virtually limited to the PIM and PSM stages. This extension will reduce the conceptual distance between requirements and other NL-based documents on the application domain on the one hand side and UML-like, project-specific models (PIM's) on the other. The semi-automatic mapping from KCP glossaries to UML models provides an automated transformation from CIM to PIM. It extends the MDA approach for use in the early software development phases. Moreover, our OBSE approach does not only take (project-specific) requirements into account but also (project-independent) ontologies.

This way, the scope of CIM's is extended to domain-spanning ontologies and future (mostly automated) MDA transformation chains may lead the developers from early-phase documents describing requirements and domain knowledge in glossary form through various model stages down to executable programs in some common programming language. This opens a way to combine Knowledge and Software Engineering and to make domain-specific knowledge via CIM's and glossaries reusable for professional SE projects.

## References

- [C-P 99] St. Cranefield and M. Purvis: A UML profile and mapping for the generation of ontology-specific content languages. In: The Knowledge Engineering Reviews, Vol. 17.1., pp. 21-39, Cambridge Univ. Press 1999
- [FKM+00] G. Fliedl, Ch. Kop, H. C. Mayr, W. Mayerthaler, Ch. Winkler: Linguistically based requirements engineering - The NIBA project. In: Data & Knowledge Engineering, Vol. 35, pp. 111 – 120 (2000)
- [GDD 06] D. Gasevic, D. Djuric, V. Devedzic: Model Driven Architecture and Ontology Development. Springer 2006
- [G-L 02] M. Gruninger, J. Lee: Ontology - Applications and Design. CACM 45.2, pp. 39-41 (2002)
- [Gru 95] T. Gruber: Towards principles for the design of ontologies for knowledge sharing, Int. J. of Human-Computer Studies 43 (1995), also: What is an Ontology?  
<http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>
- [Gua 98] N. Guarino: Formal Ontology and Information Systems. In: Proc. FOIS '98, Trento (Italy), pp 3-15. IOS Press Amsterdam 1998

- [Hes 96] W. Hesse: Theory and practice of the software process - a field study and its implications for project management; in: C. Montangero (Ed.): Software Process Technology, 5th European Workshop, EWSPT 96, pp. 241-256. LNCS 1149, Springer 1996
- [Hes 97] W. Hesse: Improving the software process guided by the EOS model. In: Proc. SPI '97 European Conference on Software Process Improvement. Barcelona 1997
- [Hes 02] W. Hesse: Das aktuelle Schlagwort: Ontologie(n). Informatik Spektrum 25.6, pp. 477-480 (2002)
- [Hes 03] W. Hesse: Dinosaur Meets Archaeopteryx? or: Is there an Alternative for Rational's Unified Process? Software and Systems Modeling (SoSyM) Vol. 2. No. 4, pp. 240-247 (2003)
- [Hes 05] W. Hesse: Ontologies in the Software Engineering process. In: R. Lenz et al. (Eds.): EAI 2005 - Tagungsband Workshop on Enterprise Application Integration, GITO-Verlag Berlin 2005 and: <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-141/>
- [Hes 06] W. Hesse: Modelle - Janusköpfe der Software-Entwicklung - oder: Mit Janus von der A-zur S-Klasse. Proc. Modellierung 2006, pp. 99-114. GI-LNI P-82, Springer 2006
- [HKM+ 04] W. Hesse, R. Kaschek, H.C. Mayr, B. Thalheim: Ontologien in der und für die Softwaretechnik. Proc. Modellierung 2004, Marburg, pp. 269-270. GI-LNI P-45, Springer 2004
- [KFM+ 05] Ch. Kop, G. Fliedl, H.C. Mayr, M. Hölbling, Th. Horn,: Extended Tagging as a Source for Mapping Requirements Texts to Conceptual Models. In: Proc. 10th Int. Conf. on Natural Language Applications for Information Systems NLDB2005, Alicante, LNCS Springer 2005
- [K-M 03] Ch. Kop, H.C. Mayr: An Interlingua based Approach to Derive State Charts form Natural Language Requirements In: Hamza M.H. (Hrsg.): Proceedings of the 7th IASTED International Conference on Software Engineering and Applications, pp. 538 – 543. ACTA Press 2003
- [KMZ 04] Ch. Kop, H.C. Mayr, T. Zavinska: Using KCPM for Defining and Integrating Domain Ontologies. Proc. Int. Workshop on Fragmentation versus Integration - Perspectives of the Web Information Systems Discipline, Brisbane Australia. LNCS, Springer 2004
- [KVH+05] Ch. Kop, J. Vöhringer, M. Hölbling, Th. Horn, Ch. Irrasch, H.C. Mayr: Tool Supported Extraction of Behavior Models. In: R.K. Kaschek et al. (Eds.): Proc. 4th Int. Conf. on Information Systems Technology and its Applications ISTA2005; Palmerston North (NZ), LNI Springer 2005
- [L-M 05] Jean-Michel Lemieux, Jeff McAffer: Eclipse Rich Client Platform: Designing, Coding, and Packaging Java™ Applications. Addison Wesley 2005
- [M-K 02] H.C. Mayr, Ch. Kop: A User Centered Approach to Requirements Modeling, Proc. Modellierung 2002, pp. 75-86. LNI p-12, Springer 2002
- [Miz 98] R. Mizoguchi: Tutorial on Ontological Engineering, Osaka University 1998  
<http://www.ei.sanken.osaka-u.ac.jp/pub/miz/Part1-pdf2.pdf>
- [N-H 89] G.M. Nijssen, T.A. Halpin: Conceptual Schema and Relational Database Design – A fact oriented approach. Prentice Hall Publ. Comp, 1989
- [OMG 03] Object Management Group (OMG): MDA Guide Version 1.0.1, <http://www.omg.org/> (2003)
- [Rus 07] A. Ruß: Übersetzung von UML-Diagrammen für die Ontologie-basierte Software-Entwicklung. Diploma thesis. Univ. Marburg 2007
- [SMK 04] A. Salbrechter, H.C. Mayr, Ch. Kop: Mapping Pre-designed Business Process Models to UML In: Hamza M.H. (Hrsg.): Proc. of the 8th IASTED International Conference on Software Engineering and Applications, pp. 400-405. ACTA Press Cambridge (USA) 2004
- [V-M 05] J. Vöhringer, H.C. Mayr: Integration of schemas on the pre-conceptual level using the KCPM-approach. Proc. 16th Int. Conference on Information Systems Development ISD2005. LNCS Springer 2005

# Viewpoint-based Meta Model Engineering

Stephan Kurpjuweit, Robert Winter

Institute of Information Management  
University of St. Gallen  
Mueller-Friedberg-Strasse 8  
9000 St. Gallen, Switzerland  
stephan.kurpjuweit@unisg.ch  
robert.winter@unisg.ch

**Abstract:** Work systems are complex artifacts that address the concerns of a large and diverse group of stakeholders. These concerns must be reflected in the models which are created as used in the development process. Current work systems engineering methods assume that concerns are more or less mutually independent and can be addressed sequentially. We argue - in analogy to other engineering disciplines - that this assumption is too restrictive. To facilitate the creation of models that simultaneously express multiple stakeholder concerns, we propose an approach which systematically elicits the stakeholder concerns, and derive a customized meta model from these concerns. We also show how this approach has been applied in an industrial case study, and propose a set of extensions to the method engineering meta model that allow method engineers to include stakeholder concerns in work system design methods.

## 1 Introduction

Models play a pivotal role in information systems and work systems<sup>1</sup> engineering: Among other purposes, models of the system under construction serve as a blueprint for its implementation, to reason about its prospective properties, to structure its development process, to decompose the system into mutually independent sub problems and to communicate it among the various stakeholders in the development process.

Like almost any engineered artifact, work systems are inherently complex and must address the concerns of a large and diverse group of stakeholders. These include participants in the design and implementation process of the work system as well as stakeholders concerned with the properties of the work system to be implemented. The models of the work system created throughout its development process must adequately

---

<sup>1</sup> Following the argumentation of Alter, we prefer the term *work system (WS)* over the more specific term *information system (IS)*. A work system is defined as “a system in which human participants and/or machines perform work using information, technology, and other resources to produce products and/or services for internal or external customers” [A106b]. Information systems can thus be seen as a specific subtype of work systems [A103], [A106a]. Therefore, throughout this paper we refer to the results of the design process as work systems.

reflect the concerns of these various stakeholders. The applied modeling concepts must appropriately express these concerns.

Most approaches in information systems and work systems engineering put concerns on the same level with process phases and the artifacts created within these phases. This reflects the assumption that concerns are more or less mutually independent and can thus be addressed one by one in sequential order (e.g. [FS95, Sc01, Wi03]). Sutton and Rouvellou [SR01] argue that this view is too restrictive because most concerns cut across process phases and the corresponding artifact types. Although this observation has been made for the domain of software engineering, it seems to be reasonable to assume that it holds true for the even broader domain of work systems engineering.

Meta models define the modeling concepts that can be used to describe models [KLC05]. Meta models can thus be seen as models of modeling languages [Fa05] and “the task of creating a meta model is the task of creating a language that is capable to describe the relevant aspects of a subject under consideration that are of interest for the future users of the created models” [Hö07].

To summarize: Innovative engineering approaches will address increasingly complex artifacts (work systems instead of information systems) by means of models that simultaneously express multiple crosscutting stakeholder concerns. Consequently, also the applied modeling concepts and meta models will be more complex and should be constructed systematically. Though a large theoretical foundation is available in the area of conceptual modeling and language construction (e.g. [BP06, LSS94, Mo05, STW03, We03, WW02]), only little work has been done to address the systematic construction of meta models that explicitly and comprehensibly represent the concerns of the various stakeholders.

In this paper we propose a systematic and applicable approach to elicit the concerns and the information needs from all stakeholders of a work system and to derive a customized meta model from these concerns and needs. Our approach incorporates and complements existing approaches and insights from the available theoretical body of knowledge. It has been applied and iteratively refined in case studies with industry partners.

The paper is structured as follows: Sections 2 and 3 discuss key concepts that lead to requirements or solution ideas for our approach. Section 4 summarizes the requirements for viewpoint-based work systems engineering. Section 5 presents our approach to meta model engineering. Section 6 discusses how our approach can be integrated into the method engineering meta model, and section 7 briefly describes an industrial application of the proposed approach.

## **2 Models, Meta Models, Stakeholders, Concerns, and Viewpoints**

According to Stachowiak [St73], a model possesses three essential properties: the representation property (a model represents an original, e.g. the work system under consideration), the reduction property (a model represents a relevant subset of all



possible properties of the original), and the pragmatic property (a model serves a purpose). Although many possible modeling purposes have been discussed, three main categories can be identified (cf. [LHM95]):

- (1) Documentation and communication (here: to document the work system as-is and to communicate it among the stakeholders)
- (2) Analysis and explanation (here: to analyze how the work system performs with respect to certain concerns and to identify strategies how it may be improved)
- (3) Design (here: to prescribe a to-be blueprint of the work system).

A model is created by a modeler and interpreted by one or more users with respect to a certain purpose [Le04]. As modelers and users of a model are not necessarily identical, it is important to ensure that both parties are able to understand the model.

Models conform<sup>2</sup> to meta models. Meta models define the modeling concepts that can be used to describe models [KLC05]. A meta model is thus a model of a modeling language [Fa05]. As a meta model itself is a model, it may conform to a meta meta model. Though in this way a hierarchy of models and meta models can be carried to the n<sup>th</sup> level, in practice the definition of the meta meta model is usually reflexive [Hö07].

According to Harel and Rumpe [HR00], a modeling language has syntax (defining the notational aspects) and semantics (defining the meaning). Additionally Kühn introduces the notation as explicit representation of the language elements [Kü04]. In this view a meta model defines the abstract syntax of a modeling language (i.e. the modeling constructs and valid ways to combine them [Hö07]), while the notation defines the concrete syntax [Di03].

As mentioned before, work systems are inherently complex and must address the concerns of a large and diverse group of stakeholders. These include systems architects, project managers, sponsors, implementers, and change agents who are participants of the design and implementation process, as well as customers, employees, managers, system operators, outsourcing partners, or the workers' council which are stakeholders concerned with the properties of the implemented work system<sup>3</sup>. Catalogs of – mostly technical – concerns have been published for software and information systems engineering (cf. [Al00, Ba04, CE00]). These include quality concerns like security (cf. [CE00]) or system performance (cf. [Al00]) as well as design related concerns like the structure and representation of data (cf. [CE00]). In the context of work systems engineering, also strategic and organizational concerns like business service realization and business process efficiency should be considered (cf. [Do04]). Based on the definition suggested by Sutton and Rouvellou [SR01] we define a concern as a matter of interest in a work system. Accordingly, a stakeholder is defined as a person or an organization that has a concern in a work system.

---

<sup>2</sup> We agree with Bézivin [Be05]. and Favre [Fa05] who argue that the term *conforms to* should be preferred over *instance of*.

<sup>3</sup> This distinction is similar to the distinction between design time and run time concerns of a software system.

The models of the work system must adequately reflect the concerns of the various stakeholders. The stakeholders' concerns and needs impact models of the work system in two ways: First, syntax and notation of the modeling language must be appropriate for the stakeholders' educational background (internal quality). This is for example relevant if employees with a business background must be able to interpret a procedural model of the work system.

Second, the design of the work system itself (i.e. the model content) must address the requirements of stakeholders to ensure that the work systems implemented on the basis of the models satisfies their requirements (external quality). This is for example the case if stakeholders responsible for the security of a work system need to ensure that appropriate technical mechanism (e.g. firewalls, encrypted network connections) or appropriate organizational mechanisms (e.g. policies to have transactions reviewed by a second set of eyes) are in place. In the latter case, the modeling language must also provide appropriate modeling constructs to express the design decisions made to address the stakeholders' concerns. The distinction between internal and external quality originates from the ISO/IEC 9126 standard [ISO01] and has been adopted to evaluate the quality of conceptual models (cf. [Mo05]).

In software engineering and requirements engineering the concept of viewpoints has been discussed since the early 1990s (cf. [Fi92, KS92, Nu94]) to simultaneously consider multiple concerns in system description and design [Do04]. The IEEE-1471 standard for architecture description [IEE00] contains the most prominent conception of viewpoints. Despite all differences between the various notions of viewpoints that have been published, most authors agree that a viewpoint describes appropriate modeling machinery (e.g., a modeling language and/or a modeling method) to capture one or more related concerns about a system. The viewpoint definition most suited for the purpose of this paper has been given by Bayer [Ba04]: "A viewpoint covers a number of concerns and defines the information associated with the concern in the metamodel." In our approach viewpoints are a major concept to structure the stakeholders' requirements and to derive meta model fragments which satisfy these requirements.

### **3 Methods and Situational Method Engineering**

The generic structure of development processes is codified in methods. Brinkkemper defines a method as "[...] an approach to perform a systems development project, based on a specific way of thinking, consisting of directions and rules, structured in a systematic way in development activities with corresponding development product" [Br96]. The discipline of method engineering (ME) is concerned with the design, the construction, and the adaption of methods. Though most method concepts discussed in the ME discipline aim at engineering and transforming information systems [TA03], the concept of a method can also be applied at the scope of work systems [Ba05].

Based on a review of approaches to method implementation and method construction, Heym [He93] and Gutzwiller [Gu94] identified five constituent elements of methods: design activities, documents specifying design results, roles, techniques, and the meta

model of the method. Braun et al. [Br05] validated this set of concepts for the description of generic methods by analyzing twelve scientific publications in the domain of method engineering. Thus, these five elements of work system design methods can be seen as a “core” method engineering meta model (unshaded entities in figure 4). As indicated by the method engineering meta model, design results conform to the meta model, which is an integral component of a method.

Recent research in method engineering has stressed the fact that methods are generic artifacts and must be adapted to the specific project type and context factors at hand [Br96, Ha97]. [Bu07] differentiates between the “project type” (defined in terms of the initial situation and the project goals to be achieved) and the “context” (defined in terms of environmental factors that may impact the project execution) and proposes an extension to the core method engineering meta model by adding the concepts “context”, and “project type”. A “situation” is a combination of certain contexts and certain project types. A “method fragment” is a generalized concept for design activities and techniques and can be adapted to a specific situation by means of “adaption mechanisms”. Figure 4 includes these extension concepts as entities shaded in dark grey.

As entire work systems are rarely designed from scratch, methods are usually applied in transformation projects (cf. [Bu07]). This raises the need to differentiate various states of the work system (e.g., as-is vs. to-be). It should be noted that as-is and to-be models of the work system may not only differ regarding represented content, but also regarding modeling concepts applied - and thus regarding the underlying meta model. This is typically the case if the transformation project applies a new paradigm to structure the work system. E.g., new paradigms are applied when moving from an application-oriented IT landscape to a service-oriented IT landscape, or from a hierarchical organizational structure to a matrix-oriented organizational structure.

#### **4 Goals for an Approach to Meta Model Engineering**

From the description of key concepts in sections 2 and 3, the following key requirement categories for systematical, viewpoint-based work systems engineering can be derived:

- (1) The fact that meta models are constructed as integral parts of work system design methods must be reflected.
- (2) The concerns and information needs of both participants in the work system construction process as well as stakeholders concerned with the properties of the work system must be considered.
- (3) The approach must be independent of a specific meta meta model and thus independent of a specific modeling technique (cf. section 5).
- (4) As an engineering approach it should facilitate reuse of meta models, provide a design rationale for modeling decisions, and address the need to adapt meta models to specific project types and to specific context factors.

## 5 A Systematic Approach to Meta Model Engineering

The purpose of our approach is to systematically elicit the concerns and the information needs from all stakeholders of a work system and to derive a meta model from these concerns and needs. Since our approach intends to be independent of specific meta modeling methodologies and meta meta models (cf. section 4, goal 3), we construct our engineering approach around any epistemologically valid meta modeling technique<sup>4</sup> that supports modeling and integration of meta models. Researchers have proposed meta modeling techniques based on the ER Model [NKF93, STM88], Attribute Grammars [Ka89, So95], Predicate Logic [NKF93], the object-oriented modeling approach [Ju00, Kü03] or other approaches [Aj96] (cf. [BSH99]).

The basic idea of our approach is to partition the complete set of modeling requirements into viewpoints, to develop a meta model fragment for each viewpoint, and to integrate the meta model fragments into a comprehensive meta model. By that means, our approach implements three engineering principles: First, the complex modeling problem is partitioned into less complex and mutually independent sub problems (divide and conquer). Second, the meta model fragments describe encapsulated solutions, which can be reused in similar situations. Third, the purpose of meta model elements can be traced back to the modeling requirements.

Consequently, the proposed approach consists of three main steps: “Requirements Elicitation”, “Meta Model Fragment Selection or Design”, and “Meta Model Fragment Integration”. We add two auxiliary steps: “Identification of Relevant Concerns” (to gain an overview of all concerns to be addressed) and “Viewpoint Relationship Overview” (to check the set of viewpoint for completeness and consistency).

---

<sup>4</sup> Similarly to software engineering methods that are typically independent of specific programming languages.

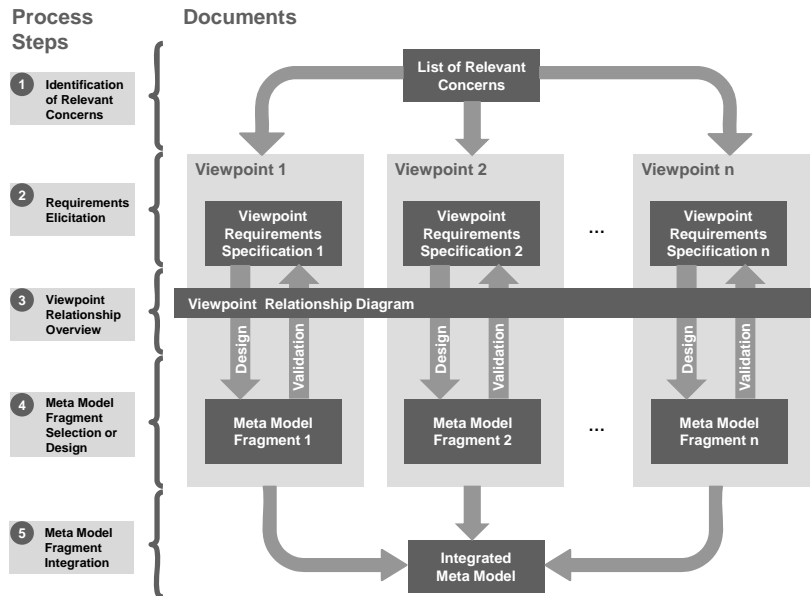


Figure 1: Viewpoint-based Meta Model Engineering – Overview

Figure 1 illustrates the process steps and documents of the proposed approach. Steps 2 and 4 are performed for each viewpoint. Thus, in a modeling project the steps can either be performed in sequential order if the scope of the meta model or the criteria to partition the requirements in viewpoints are unclear at the beginning, or the steps can be performed iteratively for each viewpoint. Our approach addresses two general application scenarios:

- (1) The initial definition of reusable viewpoint as part of a new method development project: In this case the desired degree of generality (i.e. the project types and the context factors for which the viewpoints should be applicable) must be considered.
- (2) The application of viewpoints in a concrete project: In this case the selection and integration of existing meta model fragments are the main activities.

The modeling projects we conducted with industry partners revealed aspects of both scenarios: For some concerns it was possible to reuse pre-defined meta model fragments, while for other concerns new meta model fragments had to be created.

The five steps of the proposed approach are specified in detail as follows:

### Step 1: Identification of Relevant Concerns

The goal of step 1 is to assemble a broad list of relevant concerns from a large and diverse group of stakeholders. A reference list of concerns can be used as a starting point (e.g. [AI00, CE00]). The set of concerns however should be specific for the project type

and the context factors at hand. Ideally this activity is performed within a workshop where all important stakeholders are present (or at least represented).

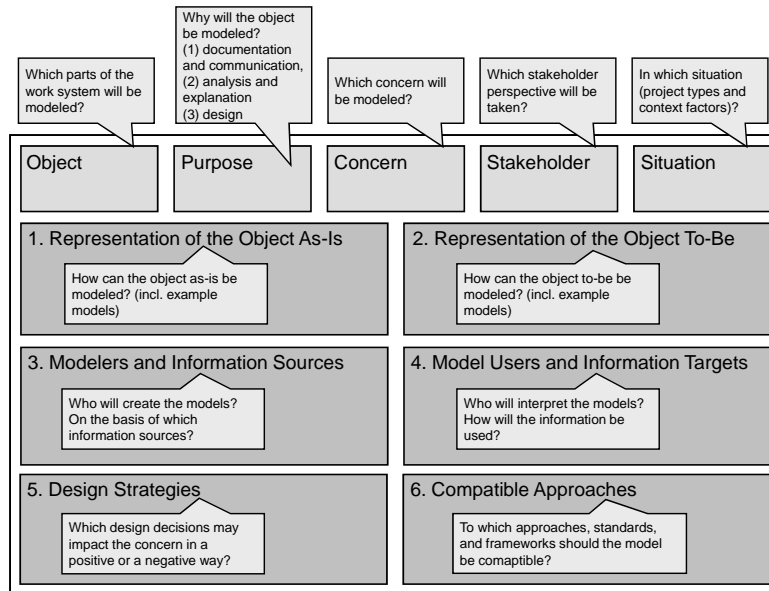


Figure 2: Viewpoint Requirements Template (VRT)

## Step 2: Requirements Elicitation

The technique applied in this step is adapted from the Goal-Question-Metric (GQM) Method [SB99], an approach to systematically derive situational metrics from questions that are in turn derived from stakeholder goals: In structured interviews with the individual stakeholders, each concern identified in step 1 is refined on the basis of the viewpoint requirements template (VRT) shown in figure 2 (a tailored version of the goal template provided by the GQM method). Related concerns may be refined together in one VRT. The VRT consists of a viewpoint goal and six additional elements to be filled in by the stakeholder. The viewpoint goal can be paraphrased: “Represent the OBJECT to {document and communicate or analyze and explain or design} the CONCERN from the perspective of STAKEHOLDER in SITUATION.” (cf. [SB99])

Next, specific questions are derived on the basis of the viewpoint goal: What questions should the model answer to achieve the viewpoint goal? These questions represent requirements regarding the information content of the models and thus the modeling concepts included in the meta model fragment.

### Step 3: Viewpoint Relationship Overview

This step is optional but may be useful to gain a model-centric overview of the method under construction and to check the information gathered in the different VRTs for correctness, completeness, and consistency. One overall viewpoint relationship diagram is created that summarizes the information about the relationships of the different viewpoints; Figure 3 illustrates the available vocabulary: “model/document type”, “modeler/model user”, “stakeholder”, “information source/target” as well as the relationships “manual transformation”, “automated transformation” (each either between two model/document types or a document/model type and an information source/target), and “association” (between stakeholder and model/document type, modeler/model user and model/document type). Figure 6 shows an example of a model relationship diagram.

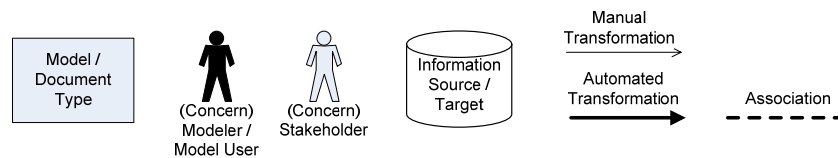


Figure 3: Viewpoint Relationship Diagram (Legend)

### Step 4: Meta Model Fragment Selection or Design

In this step the individual viewpoint specifications are complemented by adding an appropriate meta model fragment. The meta model fragment can either be designed from scratch or selected from a viewpoint catalogue. The meta model fragment must be validated against its requirements by answering the questions derived in step 2.

To design the meta model fragment from scratch, an appropriate modeling technique should be applied. As our approach intends to be independent of specific modeling techniques and meta meta models (see above), we only specify the following constraints:

- (1) The meta model must be minimal, i.e. only contain elements that are motivated by the information needs specified in the viewpoint. Otherwise effort may be spent later on to model content that cannot be interpreted with respect to a viewpoint goal (cf. [SB99]).
- (2) A design rationale for the individual meta model elements must be recorded (to address goal 4 as stated in section 4).
- (3) The semantics of the meta model elements must be clarified at least intuitively to avoid misunderstandings between different stakeholder groups.

A simple and straightforward way to achieve this is a table that provides for each meta model element a short rationale, example instances, and if necessary also negative examples that shall not be modeled as instances of the meta model element.

### **Step 5: Meta Model Fragment Integration**

Once meta model fragments for all relevant viewpoints are available, these fragments must be integrated into one integrated meta model that expresses the interrelationships between the various viewpoints. Again, the concrete approach for meta model integration depends on the underlying meta meta model. Researchers have published several approaches to meta model integration (e.g. [BSH99, ES06, Kü03, RR01]). In general, the following issues must be addressed: (a) Terminology must be adjusted to ensure that semantically similar concepts have the same name and that semantically different concepts have different names (cf. [ES06, RR01]). (b) Generalizations must be created if two concepts have similar semantics but different structures (cf. [RR01]). (c) Specializations must be created if one concept is a specialization of another concept (cf. [RR01]). (d) If the same information content is represented in different ways, such redundancies need to be removed (cf. [RR01]). (e) In order to relate meta model fragments, interface modeling concepts may have to be introduced (cf. [ES06, Kü03]).

In order to ensure that all concerns and information needs are covered, the integrated meta model should also be validated against the questions noted in the individual viewpoint specifications and against the viewpoint relationship diagram.

## **6 Extensions to the Method Engineering Meta Model**

The extended method engineering meta model proposed in [Bu07] (cf. section 3) does not reflect the viewpoint-based design of meta model fragments as presented in the paper at hand: While design activities and techniques are considered as method fragments that constitute the building blocks of methods and that can be configured and composed according to the requirements of specific project types and context characteristics, the meta model is still treated as a monolithic artifact.

To reflect the viewpoint-based meta model engineering approach discussed in the previous sections, we propose another set of extensions to the method engineering meta model by adding the following concepts:

- “stakeholder” who has one or more concerns and who might hold one or more roles in the development project.
- The stakeholder’s concerns are addressed by “design strategies” which are applied in the design activities.
- A “meta model fragment” provides the modeling concepts to capture the design decisions which are based on a design strategy in order to address a concern. Thus, meta model fragments are concern-related. The complete meta model of a method is an integration of all relevant meta model fragments as described in section 5.
- As defined in section 2, “viewpoints” package one or more concerns together with related meta model fragments.
- The concept “notation” introduces the differentiation between the abstract and the concrete syntax as proposed by Kühn [Kü04] (cf. section 2).





As indicated by the method engineering meta model, the concept of a viewpoint is also a method fragment and can thus be adapted to specific situations. This is necessary because the concerns of a stakeholder and the design strategies to address these concerns may depend on context factors and project types. For example, the concerns of the workers' council will depend on the size and the economic situation of the enterprise. Another example is that available design strategies to optimize the alignment between business processes and available IT functionalities will be different in context of a standard software as opposed to a best of breed IT strategy. Considering viewpoints as method fragments as well as composing meta models from concern-related meta model fragments are aids to overcome the monolithic approach to meta modeling which dominates traditional method engineering.

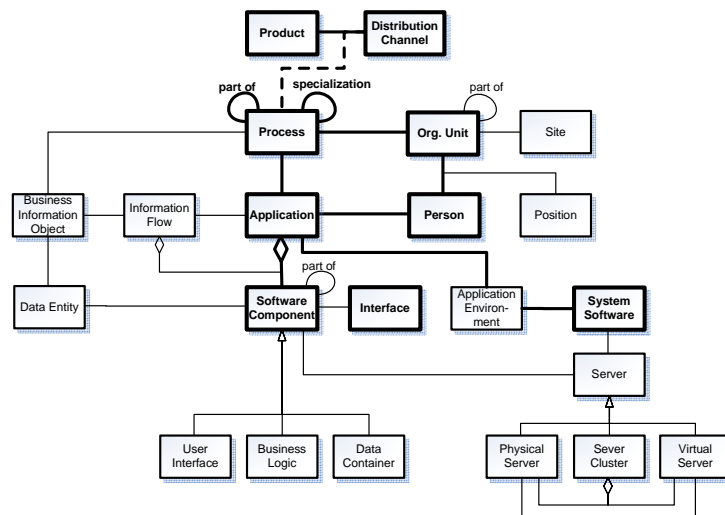


Figure 5: Complete Meta Model (simplified)

The proposed approach can be seen as a meta method, i.e. a method to engineer meta models as part of methods. In this light, the method engineering meta model presented in figure 4 becomes the meta model of our meta method, and the design results incorporated in the meta method conform to this meta model.

The appendix contains further case study material that exemplifies intermediate documents which were created throughout the meta modeling project.

## 8 Conclusion and Further Research

In our paper we presented a new approach to meta model engineering: By means of a five step process, the modeling requirements from all stakeholders of a work system are elicited, specified as viewpoints, and refined into meta model fragments which are in turn integrated into a comprehensive meta model. In this way meta models are constructed that simultaneously reflect the concerns of multiple stakeholders. Such meta models will be an important component of innovative work system design methods. Furthermore, we proposed extensions to the method engineering meta model that allow the method engineer to include stakeholder concerns in descriptions of work system design methods.

In an industrial case study our approach turned out to be useful to construct meta models which address multiple stakeholder concerns. Further research should focus on an evaluation of the proposed approach as part of the design research process. For such an evaluation we will conduct further case studies (to evaluate that the concerns of various stakeholders can be elicited and reflected properly) as well as experiments (to evaluate that the integration of meta model fragments and the verification of the integrated meta model lead to reproducible results). Based on the initial contribution presented in this paper, there are three broad directions to extend this work: First of all, a handbook of viewpoints for work system design can be assembled on the basis of existing research results from concern-focused research communities and on the basis of further industrial case studies. Second, the mechanisms to adapt viewpoints to specific project types and context factors can be formalized. This could for example be achieved by incorporating approaches from reference modeling (e.g. [Be02, Br03]). Third, our approach could be extended to provide concrete guidelines for the design and integration of meta model fragments on the basis of specific meta models. This requires the evaluation and integration of existing meta modeling techniques (e.g. [NKF93, STM88]).

## References

- [Aj96] *Ajisaka, Tsuneo*: The software quark model: a universal model for CASE repositories. In: Information and Software Technology 38 (1996), S. 173-180.
- [Al00] *Aldrich, J.*: Challenge Problems for Separation of Concerns. OOPSLA 2000 Workshop on Advanced Separation of Concerns, Minneapolis, USA, 2000.
- [Al03] *Alter, Steven*: 18 Reasons Why IT-reliant Work Systems Should Replace "The IT Artifact" as the Core Subject Matter of the IS Field. In: Communications of the Association for Information Systems 12 (2003) 23, S. 366-395.
- [Al06a] *Alter, Steven*: Work Systems and IT Artifacts - Does the Definition Matter? In: Communications of the Association for Information Systems 17 (2006) 14, S. 299-313.
- [Al06b] *Alter, Steven*: The Work System Method. 1, Work System Press, Larkspur, CA 2006.
- [Ba04] *Bayer, Joachim*: View-based Software Documentation. Ph.D. Thesis, Universität Kaiserslautern, Kaiserslautern, 2004.

- [Ba05] *Baumoel, Ulrike*: Strategic Agility through Situational Method Construction. In: *Reichwald, Ralf; Huff, Anne Sigismund (Hrsg.)*: Proceedings of the European Academy of Management Annual Conference (EURAM2005). <http://www.euram2005.de>, 2005,
- [Be02] *Becker, Joerg; Algermissen, Lars; Delfmann, Patrick; Knackstedt, Ralf*: Referenzmodellierung. In: *Das Wirtschaftsstudium* 30 (2002) 11, S. 1294-1298.
- [Be05] *Bézivin, J.*: On the Unification Power of Models. In: *Software and System Modeling* 4 (2005) 2, S. 171-188.
- [BP06] *Becker, Jörg; Pfeiffer, Daniel*: Konzeptionelle Modellierung – ein wissenschaftstheoretischer Forschungsleitfaden. In: *Lehner, Franz; Nösekabel, Holger; Kleinschmidt, Peter (Hrsg.)*: Multikonferenz Wirtschaftsinformatik (MKWI 2006), Band 2, 2006.
- [Br03] *vom Brocke, J.*: Referenzmodellierung - Gestaltung und Verteilung von Konstruktionsprozessen. Ph.D. Thesis, Logos, Berlin, 2003.
- [Br05] *Braun, Christian; Wortmann, Felix; Hafner, Martin; Winter, Robert*: Method Construction - A Core Approach to Organizational Engineering. In: *Haddad, Hisham; Liebrock, Lorie M.; Omicini, Andrea; Wainwright, Roger L. (Hrsg.)*: Proceedings of the 20th Annual ACM Symposium on Applied Computing (SAC 2005). ACM, Santa Fe 2005, 1295-1299.
- [Br96] *Brinkkemper, Sjaak*: Method Engineering - Engineering of Information Systems Development Methods and Tools. In: *Information and Software Technology* 38 (1996), S. 275-280.
- [BSH99] *Brinkkemper, Sjaak; Saeki, Motoshi; Harmsen, Anton Frank*: Meta-Modelling Based Assembly Techniques for Situational Method Engineering. In: *Information Systems* 24 (1999) 3, S. 209-228.
- [Bu07] *Bucher, Tobias; Klesse, Mario; Kurpjuweit, Stephan; Winter, Robert*: Situational Method Engineering - On the Differentiation of "Context" and "Project Type". IFIP WG8.1 Working Conference on Situational Method Engineering (ME07), 2007.
- [CE00] *Czarnecki, K.; Elsenecker, U.*: Generative Programming: Methods, Tools, and Applications. Addison-Wesley, 2000.
- [Di03] *Dijkman, Remco M.; Quartel, Dick A.C.; Pires, Luís Ferreira; Svan Sinderern, Marten J.*: An Approach to Relate Viewpoints and Modeling Languages. Seventh IEEE International Enterprise Distributed Object Computing Conference, 2003.
- [ES06] *Emerson, Matthew; Sztipanovits, Janos*: Techniques for Metamodel Composition. The 6th OOPSLA Workshop on Domain-Specific Modeling, OOPSLA, 2006.
- [Fa05] *Favre, Jean-Marie*: Foundations of Meta-Pyramids: Languages vs. Metamodels - Episode II: Story of Thotus the Baboon. In: *Bézivin, J.; Heckel, R. (Hrsg.)*: Language Engineering for Model-Driven Software Development. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Dagstuhl 2005,
- [Fi92] *Finkelstein, A.; Kramer, J.; Nuseibeh, B.; Finkelstein, L.; Goedicke, M.*: Viewpoints: a framework for integrating multiple perspectives in system development. In: *International Journal on Software Engineering and Knowledge Engineering* 2 (1992) 1, S. 31-58.
- [FS95] *Ferstl, Otto K.; Sinz, Elmar J.*: Der Ansatz des Semantischen Objektmodells (SOM) zur Modellierung von Geschäftsprozessen. In: *Wirtschaftsinformatik* 37 (1995) 3, S. 209-220.
- [Gu94] *Gutzwiller, Thomas*: Das CC RIM-Referenzmodell für den Entwurf von betrieblichen, transaktionsorientierten Informationssystemen. Physica, Heidelberg 1994.
- [Ha97] *Harmsen, Frank*: Situational Method Engineering. Moret Ernst & Young Management Consultants, Utrecht 1997.
- [He93] *Heym, Michael*: Methoden-Engineering - Spezifikation und Integration von Entwicklungsmethoden für Informationssysteme. Ph.D. Thesis, University of St. Gallen, 1993.

- [Hö07] *Höffner, Peter*: Achieving Business Process model Interoperability Using Metamodels and Ontologies. In: Österle, Hubert; Schelp, Joachim; Winter, Robert (Hrsg.): 15th European Conference on Information Systems (ECIS 2007), 2007.
- [ISO01] *ISO/IEC*: Standard 9126: Software Product Quality. International Standards Organization (ISO, International Electrotechnical Commission (IEC), 2001.
- [Ju00] *Junginger, Stefan; Kühn, Harald; Strobl, Robert; Karagiannis, Dimitris*: Ein Geschäftsprozessmanagement-Werkzeug der nächsten Generation. In: *Wirtschaftsinformatik* 42 (2000) 5, S. 392-401.
- [Ka89] *Katayama, Takuya*: A Hierarchical and Functional Software Process Description and its Enaction. 11th Int. Conf. on Software Engineering, 1989.
- [KLC05] *Klint, Paul; Lämmel, Ralf; Verhoef, Chris*: Towards an engineering discipline for grammarware. In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 14 (2005) 3, S. 331-380.
- [KS92] *Kontonya, G.; Sommerville, I.*: Viewpoints for requirements definition. In: *IEEE/BCS Software Engineering Journal* 7 (1992) 6, S. 375-387.
- [Kü03] *Kühn, Harald; Bayer, Franz; Junginger, Stefan; Karagiannis, Dimitris*: Enterprise Model Integration. In: *Bauknecht, K.; Tjoa, A. M.; Quirchmayer, G.* (Hrsg.): Fourth International Conference EC-Web 2003 – DEXA 2003, Berlin et al., 2003.
- [Kü04] *Kühn, Harald*: Methodenintegration im Business Engineering. Ph.D. Thesis, Universität, Wien, 2004.
- [Le04] *Leist, Susanne*: Methoden zur Unternehmensmodellierung - Vergleich, Anwendungen und Diskussionen der Integrationspotenziale. Habilitation, Institut für Wirtschaftsinformatik, Universität St. Gallen, St. Gallen, 2004.
- [LHM95] *Lehner, Franz; Maier, Ronald; Hildebrand, Knut*: *Wirtschaftsinformatik: Theoretische Grundlagen*. 1, Carl Hanser Verlag, München, Wien 1995.
- [LSS94] *Lindland, Odd Ivar; Sindre, Guttorm; Solvberg, Arne*: Understanding Quality in Conceptual Modeling. In: *IEEE Software* 11 (1994) 2, S. 42-49.
- [Mo05] *Moody, Daniel L.*: Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. In: *Data & Knowledge Engineering* 55 (2005) 3, <http://dx.doi.org/10.1016/j.datak.2004.12.005>, S. 243-276.
- [NKF93] *Nuseibeh, B.; Kramer, J.; Finkelstein, A.*: Expressing the relationship between multiple view in requirements specification. 15th Int. Conf. on Software Engineering, 1993.
- [Nu94] *Nuseibeh, B.*: A Multi-Perspective Framework for Method Integration. Ph.D. Thesis, University of London, London, 1994.
- [RR01] *Ralyté, Jolita; Rolland, Colette*: An Assembly Process Model for Method Engineering. In: *Dittrich, Klaus R.; Geppert, Andreas; Norrie, Moira C.* (Hrsg.): *Advanced Information Systems Engineering*, Berlin, 2001.
- [SB99] *van Solingen, Rini; Berghout, Egon*: *The Goal/Question/Metric Method*. 1, McGraw Hill, London 1999.
- [Sc01] *Scheer, August-Wilhelm*: *ARIS – Modellierungsmethoden, Metamodelle, Anwendungen*. 4, Springer, Berlin Heidelberg 2001.
- [So95] *Song, X.*: A framework for understanding the integration of design methodologies. In: *ACM SIGSOFT Software Engineering Notes* 20 (1995) 1, S. 46-54.
- [SR01] *Sutton, S. M.; Rouvellou, I.*: Issues in the Design and Implementation of a Concern-Space Modeling Schema. *Advanced Separation of Concerns Workshop*, Toronto, Canada, 2001.
- [St73] *Stachowiak, H.*: *Allgemeine Modelltheorie*. Springer, New York, Wien 1973.
- [STM88] *Sorenson, Paul G.; Tremblay, Jean-Paul; McAllister, Andrew J.*: The Metaview System for Many Specification Environments. In: *IEEE Software* 5 (1988) 2, S. 30-38.
- [STW03] *Shanks, Graeme; Tansley, Elizabeth; Weber, Ron*: Using Ontology To Validate Conceptual Models. In: *Communications of the ACM* 46 (2003) 10, S. 85-89.

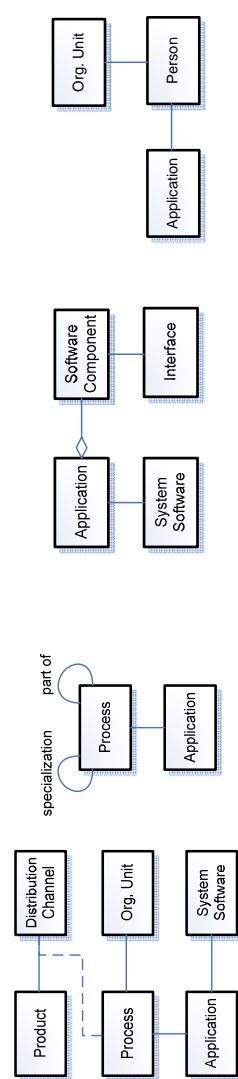
- [We03] *Weber, Ron*: Conceptual Modelling and Ontology: Possibilities and Pitfalls. In: Journal of Database Management 14 (2003) 3, S. 1-20.
- [Wi03] *Winter, Robert*: Modelle, Techniken und Werkzeuge im Business Engineering. In: *Österle, Hubert; Winter, Robert (Hrsg.): Business Engineering - Auf dem Weg zum Unternehmen des Informationszeitalters*. Springer, Berlin etc. 2003, 87-118.
- [WW02] *Wand, Yair; Weber, Ron*: Research Commentary: Information Systems and Conceptual Modeling - A Research Agenda. In: Information Systems Research 13 (2002) 4, S. 363-376.

## Appendix: Case Study Material

Table 1: Viewpoint Refinement (simplified)

Viewpoint	IT Consolidation	Business IT Alignment	Component Reuse	Ownership
<b>Object</b>	Processes, Applications	Processes, Applications	Software architecture	IT-related artifacts
<b>Purpose</b>	Analysis	Analysis	Design	Documentation
<b>Concern</b>	Cost of application operations and maintenance	Providing adequate IT for business processes	Cost of application development	Correct implementation of ownership policies
<b>Stakeholder</b>	Application architect	Process owner	Software architect	IT audit
<b>Design Strategies</b>	Consolidation of applications that are in use for a similar purposes / Consolidating of system software of the same type (e.g., DBMS, WFMS)	Providing IT functionalities for each process step / Reduction of media breaks	Reuse of software components across multiple applications / Reuse of system software (e.g., information objects, DBMS, WFMS)	Assigning explicit owners to applications and other IT-related artifacts (like components, environments, etc.)
<b>Questions</b>	Which applications are used in the individual processes (sorted by organizational unit, product, distribution channel)? / Which system software of the same type is currently in use?	Which process activities are not IT supported? / Which processes include media breaks? / Which activities are supported by multiple applications?	Which components are available in existing applications? / Which interfaces are available to use these components? / Which system software of the different types is currently in use?	Are there applications for which no owners have been defined? / Are there applications that have not been audited for more than two years?

### Meta Model Fragment



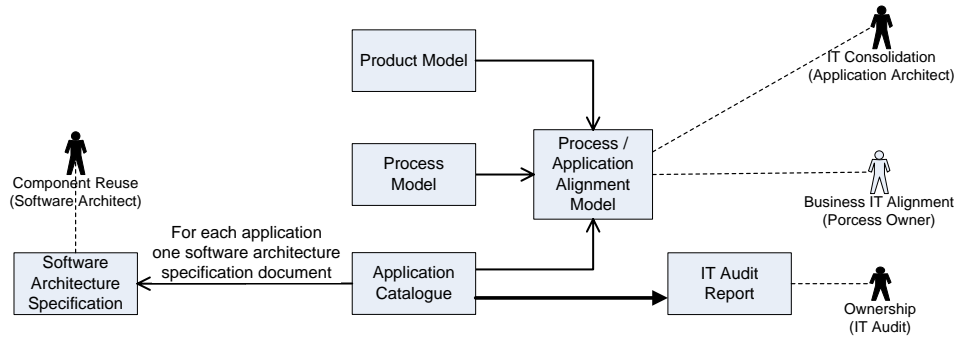


Figure 6: Viewpoint Relationship Diagram (simplified)

Table 1 illustrates the refinement of four viewpoints into meta model fragments: IT Consolidation, Business IT Alignment, Component Reuse, and Ownership. Note that for illustrative purposes these viewpoint specifications are simplified versions of the original viewpoints. The following elements of the viewpoint requirements template (VRT, cf. section 5) are omitted: representation of the object as-is, representation of the object to-be, modelers and information sources, model users and information targets. The information described in these elements is summarized in the example viewpoint relationship diagram shown in figure 6. The element “compatible approaches” of the VRT is not relevant for the viewpoints at hand. The “situation” to be addressed by the viewpoints is the architecture management and planning process of the partner company.

In all meta models shown in table 1 cardinalities and identifiers of relationships as well as attributes of classes are omitted. Figure 5 shows the integrated meta model from the four viewpoints (bold model elements) and further model elements originating from other viewpoints. All four meta model fragments are contained in the integrated meta model. The association between “application” and “system software” illustrates the need to modify meta model fragments during the integration process: Another viewpoint (Application Environment Management) raises further modeling requirements on the relationship between applications and system software that are not relevant in the context of the viewpoints “IT Consolidation” and “Component Reuse” presented here.

Figure 7 and Figure 8 illustrate how the meta model has been instantiated in models. Figure 7 shows processes and how these processes are mapped onto organizational units in a process landscape. Figure 8 shows the business IT alignment model relating processes and applications in a two dimensional matrix.



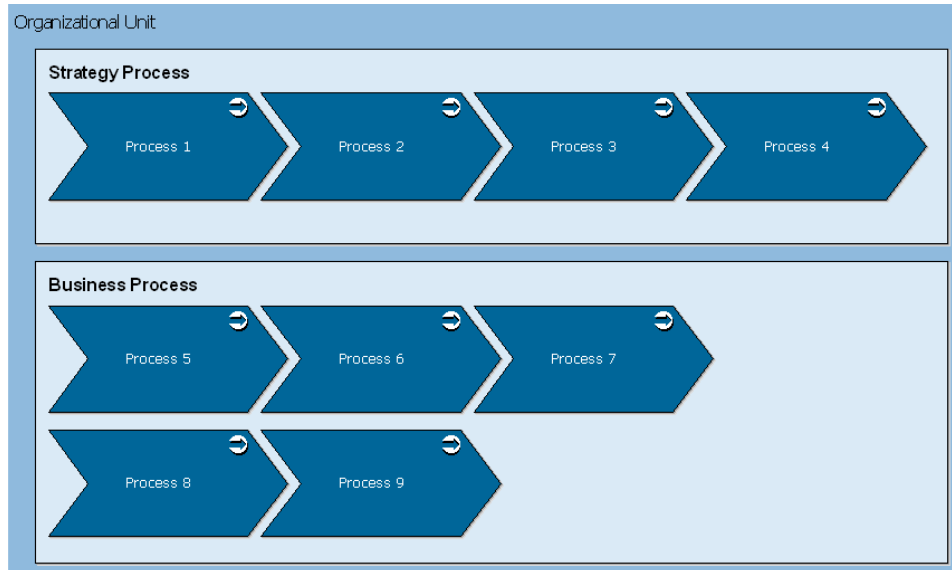


Figure 7: Example Model (Process Landscape)

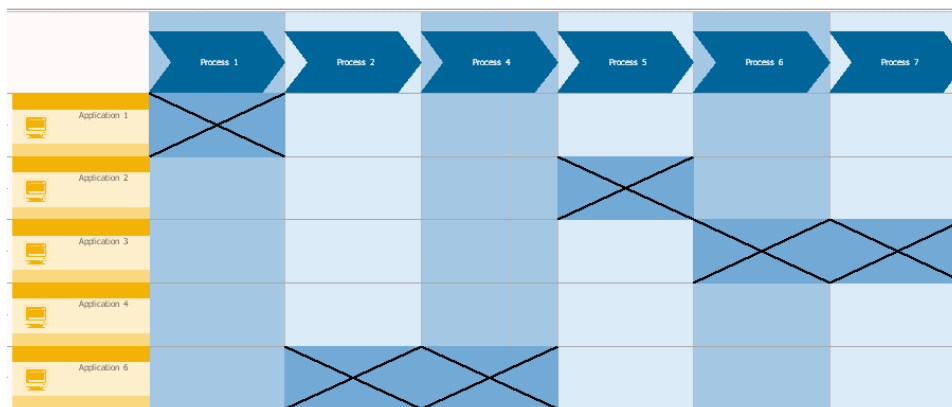


Figure 8: Example Model (Business IT Alignment Model relating Processes and Applications)



# Design and Usage of an IT-System for workplace management with ergonomic analysis under health protection aspects

Clemens Dubian

Volkswagenwerk Kassel  
Health Care / Human Resources  
clemens.dubian@volkswagen.de

Wolfgang May

Institute for Computer Science  
Göttingen University  
may@informatik.uni-goettingen.de

**Abstract:** This article describes an information system for analysis and description of workplaces under the aspects of health protection and ergonomic risks, which is currently being developed at Volkswagenwerk Kassel. The system provides an instrument for matching ergonomic risks of workplaces with work limitations of employees for an efficient assignment of employees to appropriate workplaces. It integrates data from several existing systems and collects additional data. The collection and maintenance of data is accomplished by an analysis team and by the team leaders in the factory.

Besides the functional aspects, the following two issues have been solved: minimizing the effort for the collection and maintenance of data by using a hierarchical categorization of the objects of interest and their properties. Secondly, for accomplishing the acceptance and direct benefit for the following user groups (health care, human resource management, and most of all, local supervisors), group-specific graphical user interfaces are provided.

## 1 Introduction

There are legal requirements that bind industrial businesses to document all work-specific hazards (German: Belastungen/Gefährdungen) including ergonomic risks. To accomplish activities for workplace design and for the identification of appropriate workplaces for employees with work limitations, it is necessary to get an overview of the ergonomic risks situation. Additionally, against the background of demographic change, the appropriate design of workplaces for older employees gets more and more important.

The project for developing an IT system to collect work-specific hazards is led by the departments for health care and human resource management with support from the work council. Besides the improvement of employee assignments, the system collects all other existing information of workplaces that is somehow associated with health care. Additionally, it is the pronounced aim of health care to use the system to get information about coincidences between working conditions and diseases.

This article is structured as follows: Chapter 2 explicates the notions of the application domain and describes the modeling. Chapter 3 describes relevant functionality and presents some GUIs. Chapter 4 concludes the paper with a short discussion including status and acceptance, transferability of the approach, and perspectives.

## 2 Application Domain and Modeling

### 2.1 Concepts of the Application Domain

The developed system combines information about different views on the company, especially its product division. The structural classification of the plant into *organizational units* is essential for user guidance and responsibilities:

In the Volkswagenwerk Kassel, the classification begins on the plant management level and leads over divisions to local teams. All workplaces in the production are grouped into teams, each of them led by foremen (more concretely, one foreman per shift; such a team is usually called “Meisterschaft”). This structure is not specific to the given use case, but is a common structure of large industrial plants. In this presentation we restrict ourselves to the production area; workplaces in e.g. logistics, administration and health care can be handled in a similar way.

Thus, large parts of the modeling are concerned with aspects of the production area. Volkswagen Kassel employs about 14,000 people at about 7,000 workplaces which include about 20,000 machines to analyze. Information is not captured and stored separately for each item, but managed in categories at different levels of abstraction. Conceptually this is modeled by *multiple inheritance* at different *dimensions*. The main dimensions are (i) the equipment ontology of machines, and (ii) the structuring of the actual working processes. To adequately formalize these coherences, the abstract concept of *categories* describes an “entity to analyze”. Categories inherit from other categories and add own properties. Categories represent machine classes (e.g. milling machines), machine types (e.g. Gleason Pfauter GP90 as a special type of milling machines), tasks or any other (abstract) concept related to the workflows.

From the working process structure aspect, the focus of analyzed workplaces differentiates between tasks of one job; large assembly lines are partitioned in (abstract) components. A *workplace* includes all machines and tasks handled by one employee. The assignment of machines to workplaces is weighted by percentages, so that one machine can be assigned to several workplaces.

The ergonomic risks analysis combines both views. The basic analysis objects are concrete objects such as machines, and categories. The ergonomic risks of each workplace are then derived from its composition. Besides the analysis of machines or categories, there are hazards caused by the work location such as noise or temperature, or exposure to hazardous substances.

Furthermore workplaces themselves are aggregated to workplace packages. Thus, a group of workplaces together performing a larger working process can be associated to a group of employees who organize who is assigned to which workplace, optionally including job-rotation. Each workplace and each workplace package is associated to exactly one operating team. Therefore the view of a team's foreman is one of the main views of the workplace management system. It is used by the foremen to maintain the information about the structure and the actual assignments in their area. The functionality and the GUI are described in more detail in Section 3.

In the description above, the abstract concept of "workplace" has been used for modeling the production environment. For modeling the actual production workflows, workplaces are associated to employees. For that, one function of the workplace management system is the structural association of employees to workplaces in terms of planning and scheduling. Based on that, the actual occupation of workplaces within a shift is maintained.

For the assignment of employees to workplaces, *work limitations* of employees must be taken into account: upon medical checks, the health care department states work limitations for employees, like "no heavy lifting". German legislation requires a certain system of *preventive medical checkups* under specific circumstances; this is also organized via the system.

## 2.2 Existing Information Infrastructure

The workplace management system integrates different views and data, which are partially stored in other already existing information systems that are maintained autonomously. In the following the connections from the workplace management system to other information systems are described (see Figure 1). The current system is a prototype; therefore the described connections contain manual data transfer.

Structural data contains the organizational structure of Volkswagen. Organizational units are used for defining access policies. The structural data is maintained in the SAP system by the human resource management. Personnel base data is also maintained by the human resource management within the SAP system. Current data about employees who are present on the factory site (electronical check) can be obtained from the *Access Authority System (ZUBESY)*, a plant security system. The actual realization of this connection is subject to privacy protection issues.

The *Hall Layout System (HLS)* contains layouts of every hall, including the positioning of the individual machines. The workplace management system uses the layouts to structure and associate workgroups, workplaces, machines and employees via a graphical user interface.

The *central file for operating equipment and machines (ZBM)* contains and maintains all data on machines and operating equipment including inventory number, its assignment to organizational units and the positioning with respect to the coordinates in the HLS.

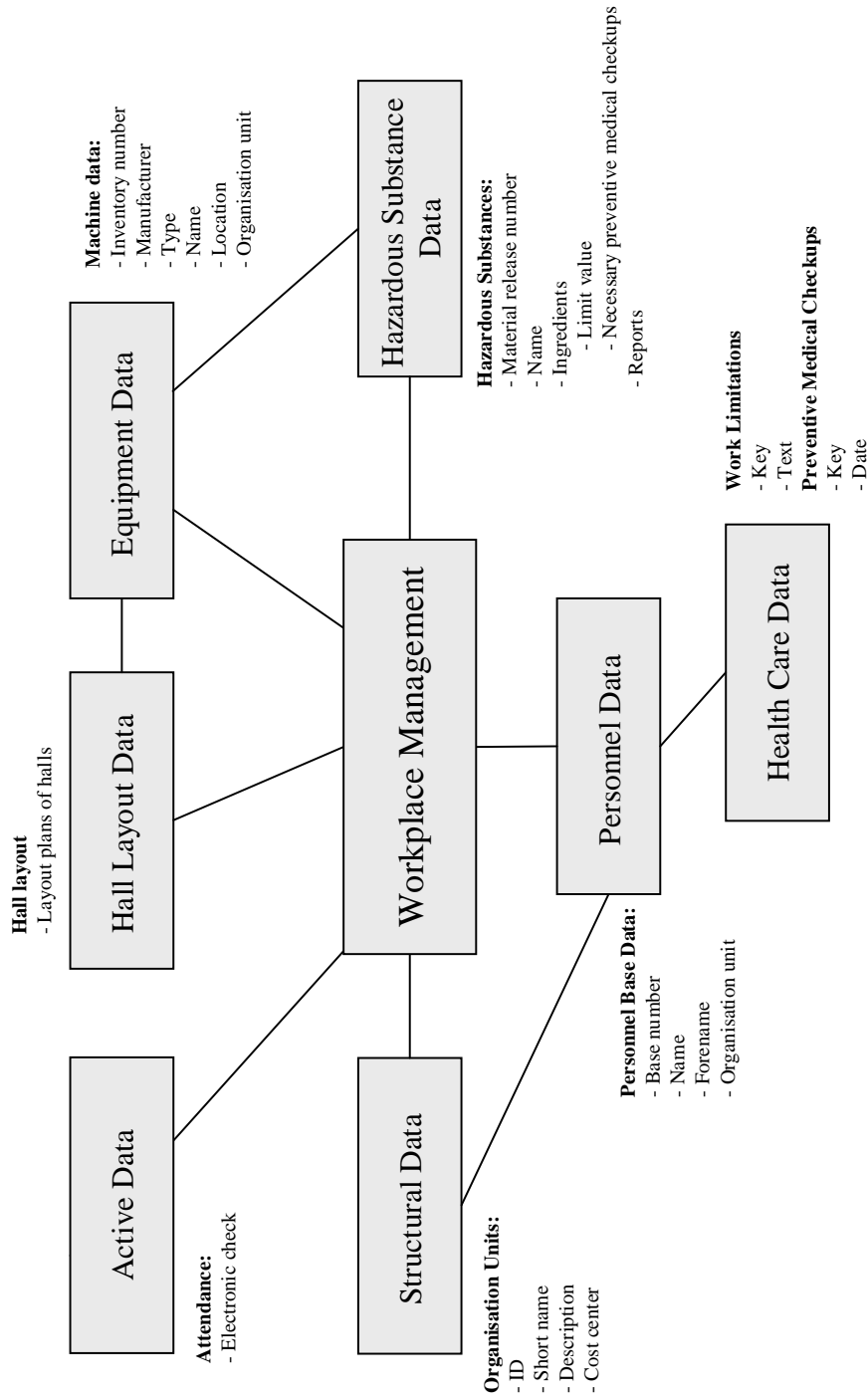


Figure 1: Connections of the workplace management system

The division for chemical safety maintains the *hazardous substances database* (APM). Hazardous substances are associated to workplaces within workplace analysis and through hazardous substance measurements.

Medical data, which is important for human resource management and team leaders, is transferred from the *Occupational Medicine Administrative System* (AMVW) of the health care division to the SAP system. The workplace management system gets the combined data through a protected interface from the SAP system.

### 2.3 The Conceptual Model

The conceptual model of the application area is shown in Figure 2, where concepts are grouped into individual-related information, workplace structure and working environment characteristics.

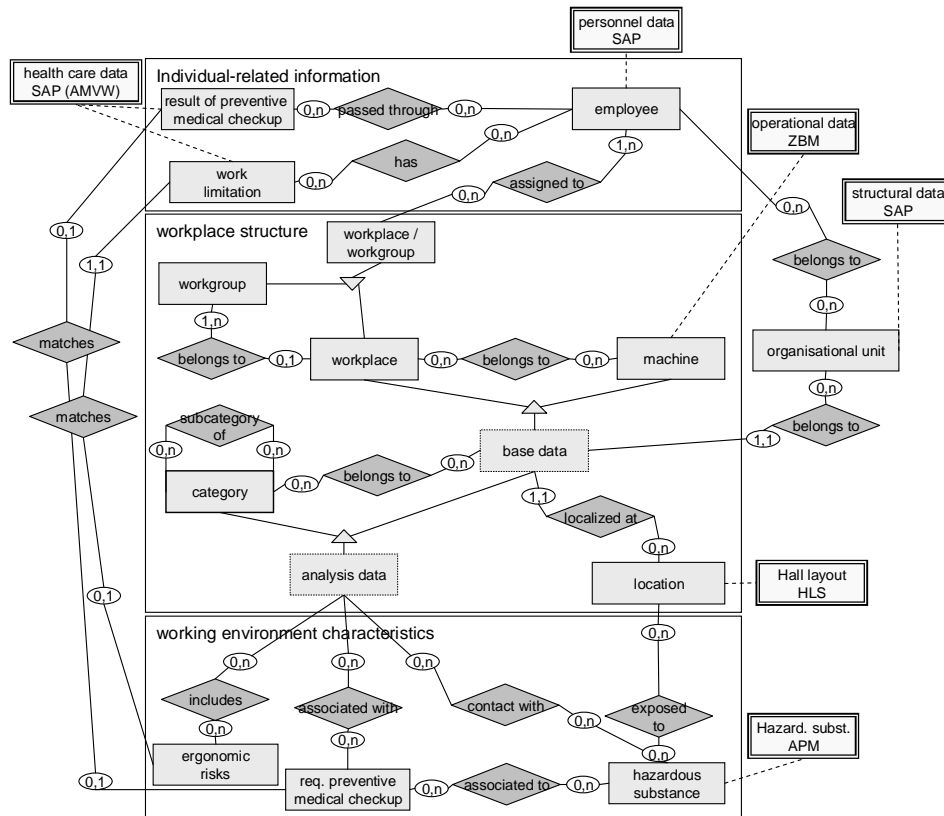


Figure 2: ER-diagram of the workplace management system

The modeling of the workplace structure mirrors the described structural aspects. *Base data* describes the combination of an instance of a category and a location inside the

plant, referring to the Hall Layout System. It combines properties of the location with properties of the category and is associated to an organizational unit (referring to the SAP system). Information about machines refers to the operational data kept in the ZBM; a more concrete example is given at the end of this section.

The analysis data about workplace environment characteristics contains ergonomic risks, exposure to hazardous substances and the required medical checkups. This information is related to each of the categories.

Personnel data and some medical data from employees are imported from the SAP system into the workplace management system. An employee can be assigned to a single workplace or to a workgroup. If an employee is associated to a workplace or workgroup, the workplace management system matches the work limitations of the employee with the ergonomic risks of the workplace and relates with preventive medical checkups.

**Example.** Input of initial data (such as the definition of the category hierarchy including the weighting and data about machine classes and types) is done by analysts. As shown in, a Gleason-Pfauter P90 is a milling machine (German: Fräsmaschine; 90%) as well as a deburring machine (10%) and therefore combines analysis of both categories. At a Gleason-Pfauter P90, usual millcutting (70%) and deburring (10%) are dry, more complex millcutting (20%) requires using cooling lubricant. Each instance is associated with its location. Each of the categories is associated with its analysis data, and the properties of the actual machine are obtained as a combination of that data.

## 2.4 Application-Specific Modeling of Risk Analysis

The core aspect of the system is the information about risk analysis and connecting it with individual-related data. *Risk analysis* is the analysis of all risks that employees are exposed to during their work time. For the given application, risk analysis consists of *ergonomic risk analysis*, which is described in more detail below, and *hazardous substance analysis*. Based on the analysis of different individual hazards of a workplace, the overall characteristics of the workplace can be summarized. For the actual assignment of employees to workplaces, these characteristics must be matched with the work limitations of the employees.

The *ergonomic risk analysis* [La04] is based on a questionnaire. The questionnaire includes general aspects of given tasks and workplaces, and complementary questions related to the kinds of potential work limitations. Risk analysis distinguishes between exposures, e.g. hazardous substances or oil, ergonomic risks related to physical strain, e.g. heavy lifting, as well as special requirements like stereoscopic vision, e.g. for driving a stacker, or work in night shift.

Risks and exposures are classified according to *how* they are characterized:

(1) A *risk or exposure without weighting* is just quantified by “yes” or “no”. It excludes employees with the corresponding work limitation. The exposure “skin contact with oil”





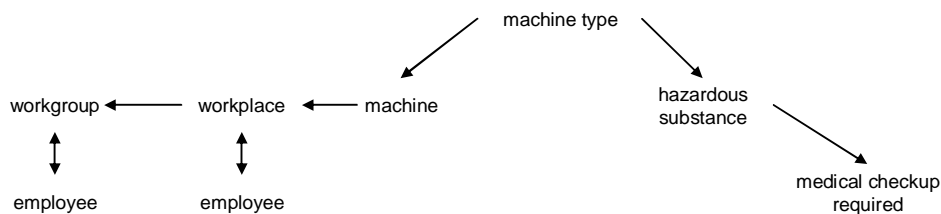
for example excludes employees who are allergic to oil no matter if the contact is for one hour or one minute.

(2) A risk or exposure with weighting is expressed in terms of the percentage of working time. Those hazards only exclude employees with the corresponding work limitation when they are associated for a specific duration. For example there could be one task to bend down to lift work pieces. If this task amounts to 2% of the working time it should be no problem even for an employee with the work limitation “no bending down often”. When forming a working group, the overall risk or exposure is computed according to the distribution; depending on the characteristics of the working group (e.g. frequency of job rotation), this can amount to maximum- or average-based formulas.

(3) For ergonomic risks or exposures that have more complex characteristics, there are Key Indicator Methods of the Federal Office for Occupational Safety and Health (Bundesanstalt für Arbeitsschutz und Arbeitsmedizin, BAuA). With help of these Key Indicator Methods the risk or exposure can be calculated and results in a score. However multiple scores from different tasks cannot be added easily. To determine a score, the interim results must be summarized in a given manner and the Key Indicator Method must be performed again with the summarized values. The result is a structural rating with multiple attributes.

Work limitations wrt. ergonomic risks or exposures are expressed in the same way by prohibitions (“no skin contact to oil”) or threshold values (“bending down must not be more than 10% of working time”).

As a consequence of the analysis, the necessity of preventive medical checkups can be derived through the category network. For instance a preventive medical checkup implied by exposition to a certain hazardous substance related to a certain machine type can be derived for employees assigned to workplaces and workgroups by the following relationships:



Categorization lowers redundancy and therefore supports consistency not only for collecting data, but especially reduces efforts for data maintenance.

### 3 Functionality

User groups in the workplace management system have access to different functionalities according to their requirements and their rights. A central task is to collect and

maintain the information about risk analysis and workflow structure. This is done by analysts and foremen. The actual assignment of employees to workplaces that changes daily is maintained locally by the foremen. The information system is also used by different user groups for several retrieval tasks. The Human Resource Department and the foremen use it for finding and assigning workplaces to employees with work limitations. The Health Care Department obtains background information about an employee's workplace conditions in the context of medical checkups. Additionally the scheduling of medical checkups is supported.

For each user group, a specific GUI is implemented, according to its needs. This chapter discusses the functionality and gives an impression of the target-group specific GUIs.

### **3.1 Maintenance of Individual-Independent Information**

The initial risk analysis is performed by a team of analysts, including ergonomists, planners, representatives of chemical safety and occupational health physicians. The risk analysis information can be updated later. The initial workplace definition is also done by the analysts. Changes in workflows are usually maintained by the foremen by changing associations and attribute values (e.g. percentages). The changes of associations, both in risk analysis and workplace definitions lead automatically to a new calculation of affected categories by the system.

The task of analysts is to build categories and to enter data about analysis objects, using a tree view-based GUI for designing the category structure. The actual machine categories are populated by importing existing data about machines from the ZBM. The analysis data is then added in cooperation between ergonomic specialists from Health Care and the foremen by using a questionnaire. The original concept of the questionnaire was taken from Frieling [Fr04]. Modifications and extensions are made by suggestions from a consortium of automotive industries as well as from an ergonomic workgroup of Volkswagen and incorporated by the Health Care department.

The second stage of the initialization consists of defining workflows and workplaces. This step is also carried out by the analysts together with the foremen, using either the tree view-based interface described above, or the graphical interface of the foremen that is described in Section 3.2.

After completing the initialization, the foremen take on maintenance of workplace definitions and the actual daily assignment of employees to workplaces.

### **3.2 Foremen View: Definition of Workplaces and Employee Assignment**

The main focus for designing this user interface was to enable foremen (i.e. people with a non-IT professional context) to handle the workplace management system without the need for long instructions and with a minimum of time for data maintenance.

At first “dot-plans” (Figure 4, right-hand side) were used to identify workplaces. These plans are abstract illustrations with rectangles as machines and dots as employees (work-places). After further interviews with foremen it turned out, that the plans of the hall layout provide a useful complementary view as shown in Figure 4, left-hand side.

Associations can be made by drag&drop, each object has an interactive context menu and double-click and mouseOver for detail information. This GUI is used for four tasks, in which different items are shown in the graphics:

**Definition of workplaces:** Machines and workplaces (A) are positioned according to their location information (note that a workplace as an abstract notion gets a virtual location). Assignments of machines to workplaces are done by drag&drop and are indicated by (blue) connection lines.

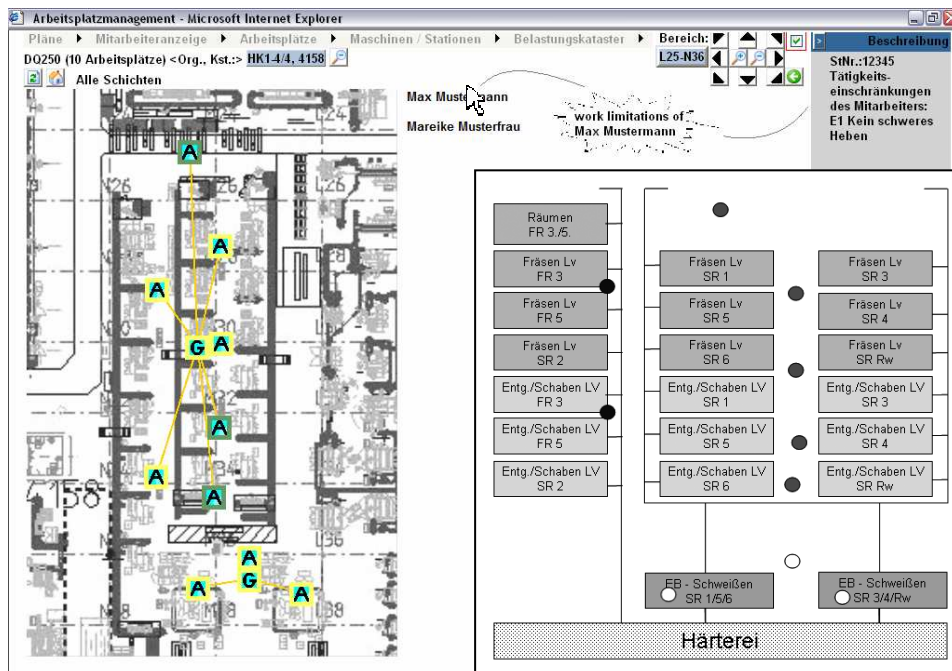


Figure 4: Foremen’s view with hall layout background

**Definition of workgroups:** Workplaces (A) and workgroups (G) are positioned according to their (virtual) location information. Assignments of workplaces to workgroups are done by drag&drop and are indicated by (yellow) connection lines (cf. Figure 4, left-hand side).

**Assignment of employees:** For each actual shift, the employees are assigned to workplaces or workgroups. The hall layout plan shows workplaces (A) and workgroups (G), and additionally all employees already present in the plant are listed to the right of the hall plan (cf. the test employee in Figure 4 where the mouse cursor points to). Moving the mouse cursor to an employee, the description area on the upper right shows work

limitations of the employee (in the above example “no heavy lifting”; in German: “Kein schweres Heben”). All workplaces and workgroups get a specific colored border according to the match between their ergonomic risks and work limitations of the employee: all workplaces are marked with green icons (A, cf. Figure 4 left-hand side), where this employee has no conflicting work limitations. In case of possibly conflicting work limitations, the workplace is marked with a yellow icon (A; this requires a decision on a by-case-basis), and if there is a definite conflict with a work limitation, the workplace is marked with a red icon. The actual assignment is again done by the foreman by drag&drop. Pointing the mouse cursor to a workplace on the other hand, all employees in the visible area get a specific colored border according to the match.

**Detailed search to assign employees with work limitations:** In that view, machines, workplaces and employees are shown. By pointing to a machine or an employee, the appropriate detailed matches are shown. Through this view, light duty work can be created by setting up a workplace definition that consists only of operations of specific machines without ergonomic risks.

To minimize the latency during maintenance, the workplace management system functions mostly without full post backs. Server functionalities are initiated with JavaScript, results integrated with AJAX [Ga05]. Therefore workplace definitions and associations of employees to workplaces can be maintained fast and efficiently.

### 3.3 View for Human Resource Management

The human resource management is interested in finding adequate workplaces for each employee. Adequate in that case means that the employee can handle the workload without limitations. This search can be invoked for an employee, or for any combination of work limitations<sup>1</sup>, starting at any height of the tree of organizational structure. Figure 5 illustrates such a search for the organizational unit HK1, which is gearbox production, with the work limitation “no heavy lifting” (“Kein schweres Heben”). The hall layout is used as background and the user can navigate through it.

If more than 50 workplaces are matched against an employee, they are combined in rectangles for each organizational unit below the chosen one, e.g. HK1-4/3, HK1-4/4 and HK1-4/5. Each rectangle shows included workplaces in numbers. These numbers are colored according to the match against the employee (e.g. in HK1-4/4 there are 17 adequate (green) workplaces, 14 possibly adequate (yellow) and no non-adequate (red) workplaces). By clicking on a rectangle, it is enlarged to get details of every included workplace. By further zooming in, the view gets more and more detailed, until it shows the same detailed view including individual workplaces as shown in Figure 4.

The search is e.g. needed in cases when employees cannot work at their former workplace or in their former workgroup anymore. This happens for example, when an em-

---

<sup>1</sup> Adequacy of a workplace for an employee also depends on his knowledge and abilities.

ployee had an accident or gets new work limitations for other medical reasons by an occupational health physician. Foremen and superiors can use the search with access limited to their organizational unit.

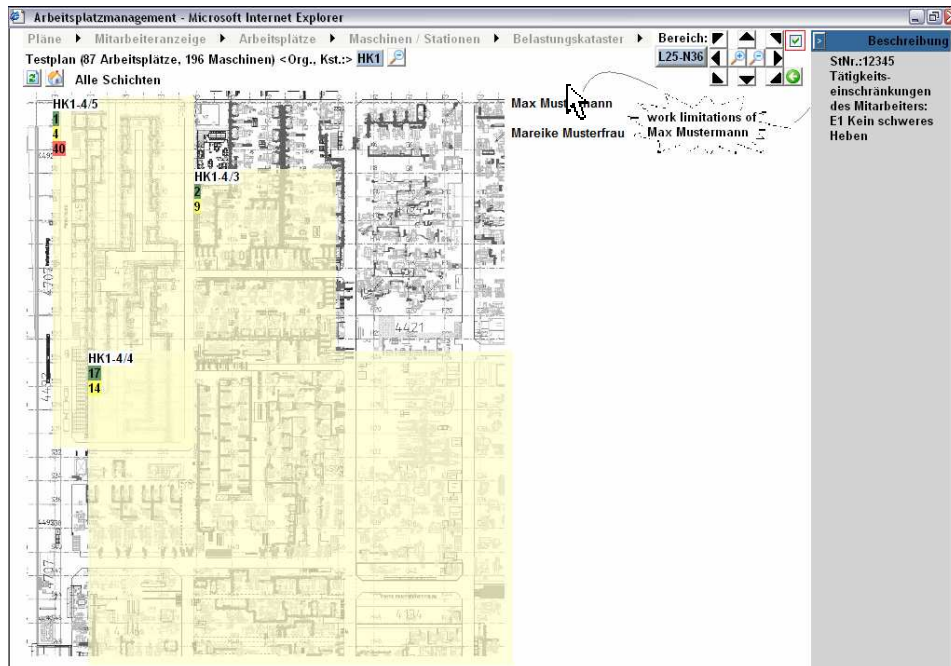


Figure 5: Search for an adequate workplace in multiple organizational units

### 3.4 View for the Health Care Department

Every time an employee consults the Health Care department, the physician can use the system to get an impression of the work situation of the employee within a few seconds: the workplace management system enables occupational health physicians to have a closer look at the specific workplace. *Workplace summarization* gives an overview of all medically relevant information of the workplace or the workgroup. Managed by a tree view, all associations including weightings and subtotals are listed. Photos and videos illustrate why specific ergonomic risks are associated to a workplace.

By keeping history of all associations of an employee to his workplaces, all working conditions the employee was exposed to during his working life are registered. In that way the so called *exposition record* is generated. Those records are a substantial fundament for investigating *coincidences between working conditions and diseases*.

## 4 Discussion

We presented a use case of enterprise modeling on a detailed level. The modeling uses a concept hierarchy for representing and structuring equipment and the actual work in the production division. The categorization is applied to capture and maintain information about ergonomic issues. In addition to the modeling and the analysis data which are captured by analysts during the project, the integration of the existing information from heterogeneous sources had to be solved on the conceptual level as shown in the diagrams, and then had to be implemented.

**Relevance.** The topic “risk analysis and work limitations of employees” and the corresponding additional expenses for searching adequate workplaces emerged recently due to a German law that implements European regulations for occupational health and safety, and due to the background of demographic change.

**Status.** The workplace management system is a prototype developed by the Health Care and Human Resource Management departments. It implements a standard three layer architecture with data layer, application layer and representation layer where the Intranet application connects to a relational database.

**Acceptance.** Critical for the benefit of every information system is its acceptance by all involved user groups. For that purpose the system, the modeling and especially the user interfaces have been designed in close communication with prospective users, incorporating their continuous feedback. As most of the maintenance and daily use is done by the foremen in the production, their GUI is a crucial factor to achieve ongoing success. Until now workplaces for approximately 800 employees are being analyzed and about 40 users are testing the usability.

**Transferability.** The described modeling of work categorization and matching possibilities is transferable to other larger industrial plants. Concerning risk analysis in particular, different environments lead to different restrictions and requirements. The model provides a framework that can be used for any matching possibility that fits into one of the three summation methods.

**Related Work.** Work on enterprise ontologies [Us95, Di06] usually deals with high-level aspects of enterprises like organizational structure, administration, business process models and patterns, business transactions, strategic goals etc. In contrast, our approach focuses on the modeling of the industrial production level.

Usually industrial workflows are designed and analyzed by using MTM [BL06] (Methods-Time Measurement) methods and tools. *MTM-Ergonomics* [MTMe] is a proposal based on the experiences of several companies. MTM-Ergonomics and similar approaches (for an overview see [La04]), in most cases proprietary, systems allow for workplace analysis and for the search for workplaces corresponding to given restrictions. However, none of the systems known to the authors integrates personnel data.

**New Features.** By integrating personnel data, the Workplace Management System allows actual matching of employees to workplaces and thus supports the daily planning and scheduling and runtime rescheduling. Another distinguishing feature is that it is based on generating an ontology in terms of a hierarchical structuring of the production. This categorization reduces maintenance efforts since work parts and machine types are analyzed separately. They allow changes in workplace definitions without the need for new analyses: if a workplace definition changes, the summarized analysis is recalculated as shown in Section 2.4.

**Perspectives.** The functionality of the current prototype focuses on the requirements of running the production process in the plant. Further on, more advanced usage of the information available in the system can include the following:

From an ergonomic point of view, the categorization can be used for testing transferability of improvements to similar machines. Data mining algorithms can be used to search for peculiar or similar structures and enlarge and refine categorization; activity recommendations could be derived automatically.

Up to now, the workplace management system includes import and gathering of required data and maintenance of analysis data as well as generating reports. The full historization of all maintained and all integrated data builds a large data stock. Thus, a powerful fundament for upcoming studies is build.

For health care, exposition files are very precious data for further analysis. The search for similarities between exposition records from employees who suffer of a specific disease could provide knowledge about coincidences. On a long term perspective, large studies about effects of health care actions or about effects of hazards can be developed with minimum effort. This also includes studies about contact to hazardous substances.

## References

- [Ga05] J.J. Garrett: Ajax: A New Approach to Web Applications. Tech. Rep. Adaptive Path, <http://adaptivepath.com/publications/essays/archives/000385.php>, 2005.
- [Fr04] E. Frieling: DFG Schwerpunktprogramm 1184: Altersdifferenzierte Arbeitssysteme. Zeitschrift für Arbeitswissenschaft, 2006; S. 71-80.
- [La04] K. Landau: Montageprozesse gestalten. ergonomia Verlag, Stuttgart, 2004.
- [BL06] R. Bokranz, K. Landau: MTM-Handbuch. Schäffer-Poeschel, Stuttgart, 2006.
- [MTMe][http://www.mtm.com/produkte/software/ticon\\_modul\\_ergo.php](http://www.mtm.com/produkte/software/ticon_modul_ergo.php)
- [Di06] Jan L. G. Dietz: Enterprise Ontology: Theory and Methodology. Springer, 2006.
- [UK95] M. Uschold, M. King, S. Moralee and Y. Zorgios: The Enterprise Ontology, <http://www.aii.ed.ac.uk/~entprise/enterprise/ontology.html>, 1995



# On Industrial Use of Requirements Engineering Techniques

Lars Bækgaard

Department of Business Studies

Aarhus School of Business, University of Aarhus

Fuglesangs Alle 4, DK-8210 Aarhus V, Denmark

[lab@asb.dk](mailto:lab@asb.dk)

Jens Bæk Jørgensen and Kristian Bisgaard Lassen

Department of Computer Science, University of Aarhus

IT-parken, Aabogade 34, DK-8200 Aarhus N, Denmark

[jbj@daimi.au.dk](mailto:jbj@daimi.au.dk)

[k.b.lassen@daimi.au.dk](mailto:k.b.lassen@daimi.au.dk)

**Abstract.** We discuss two experiments in which requirements engineering techniques has been used and evaluated. In the first experiment a technique called Executable Use Cases is applied in the development of an IT system for public utility services. In the second experiment a technique called Activity Cases is applied in the development of an IT system for a public library. For each experiment we discuss the lessons that we have learned. We use the lessons to scetch a plan for future research in terms of a set of scenarios for combined use of Executable Use Cases and Activity Cases.

## 1 Introduction

We present, discuss, and learn from two experiments in which two requirements engineering techniques called Executable Use Cases and Activity Cases has been applied and evaluated. Executable Use Cases support animation of process models and Activity Cases support flexible activity modeling. Based on the experiments we present a set of scenarios that combine the two techniques.

Development methods like Structured Analysis [De78], Multiview [AW90] and Contextual Design [BH98] are based on the assumption that information systems development should be based on a thorough understanding of the work contexts of the systems. Our work is based on the same assumption and our aim is to provide create techniques that provide such understanding and facilitate creative requirements processes that are based on well-founded understandings of work contexts.

Our research method can be characterized as design science [MS95], [HM04]. We have created the techniques and experimented with their use in order to evaluate their relevance as requirements engineering techniques.

We conducted two experiments based on result from a requirements specification workshop in which the techniques Executable Use Cases and Activity Cases were presented and discussed. The experiments were conducted in real-world requirements engineering situations. The experiments represents the first af a series of evaluation activities in our design science project. In future activities in the project we will experiment with combinations of the two techniques and we will decide if it is desirable to modify the techniques. This implies that the presented results are preliminary results from a larger ongoing project.

The paper is structured as follows. Section 2 we introduce the notion of executable use cases and discuss an experiment where a set of business processes are modeled by means of executable use cases. In Section 3 we introduce the notion of activity cases and we discuss an experiment where a set of business processes are modeled by means of activity cases. In Section 4 we discuss some of the implications of the experiments and we suggest directions for future research.

## 2 Experiment 1 - Executable Use Cases<sup>1</sup>

The *Public Utilities Aalborg (PUA)* are responsible for providing gas, electricity, heating, water, sewer services, and garbage collection to the Aalborg region, which has approximately 160,000 inhabitants. PUA is the shared administration for six companies, one for each type of service. In total, PUA has around 450 employees. The employees uses a number of IT systems in their daily work. Examples are accounting systems and geographical information systems (GIS).

PUA has hired the software company *WM-data* to deliver a new IT system, which we will call the *Authorization System (AS)*. AS will be used by PUA to administer the access rights for different users of different IT systems. AS should support creation of new user accounts, update of the rights of existing users, and closing of user accounts.

WM-data is a Scandinavian company, whose branch in Aalborg will develop AS. It is beforehand decided that AS should be based on OIM<sup>2</sup>—a standard off-the-shelf system for management of user rights and privileges across enterprises. This means that in the first place, WM-data must carry out analysis work aimed at aligning the real needs of PUA with what is actually possible with the given technology, i.e., OIM.

As part of this analysis work, WM-data has been interested in trying out the requirements engineering approach that we call *Executable Use Cases (EUCs)* [JB04b], and we have used the AS project as an experiment for our research on EUCs.

---

<sup>1</sup> Experiment 1 was carried out by the second and the third authors.

<sup>2</sup> Oracle Identity Manager, formerly known as Oracle Xellerate Identity Provisioning.

We will describe use of EUCs in the AS analysis project in this section and we will report on some preliminary findings, based on work carried out in the period April-June 2007, where a series of analysis workshops were held; the participants in the workshops were personnel from PUA, analysts and developers from WM-data, and the second and the third authors of this paper.

### Executable Use Cases (EUCs)

An *Executable Use Case (EUC)* [JB04b] supports specification, validation, and elicitation of requirements. EUCs spur communication between stakeholders and can be used to narrow the gap between informal ideas about requirements and the formalisation that eventually emerges when a system is implemented. An EUC consists of three tiers. Each tier represents the considered work processes that must be supported by a new system. The tiers use different representations: Tier 1 (the *informal tier*) is an informal description; tier 2 (the *formal tier*) is a formal, executable model; tier 3 (the *animation tier*) is a graphical animation of tier 2, which uses only concepts and terminology that are familiar to and understandable for the future users of the new system. Tier 3 has the potential to offer significant advantages as a means of communication.

The three tiers of an EUC should be created and executed in an iterative fashion. The first version of tier 1 is based on domain analysis, and the first version of tiers 2 and 3, respectively, is based on the tier immediately below. Figure 1 illustrates the approach.

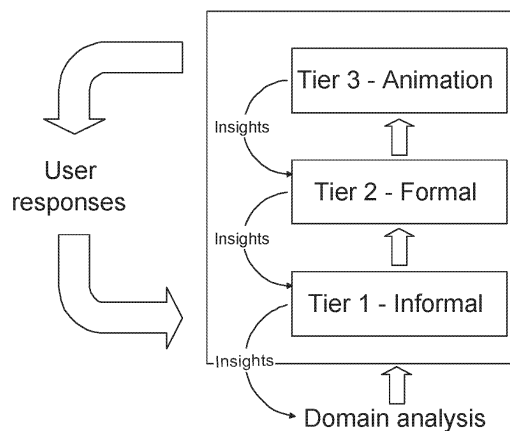


Figure 1 Executable Use Cases

EUCs have notable similarities with traditional high-fidelity prototypes of IT systems; this comparison is made in more detail in [BJ04]. In [JB04a], it is described how an EUC can be used to link and ensure consistency between, in the sense of Jackson [Ja04], user-level requirements and technical software specifications.

An EUC can have a broader scope than a traditional use case [Co01]. The latter is a description of a sequence of interactions between external actors and a system that happens at the interface of the system. An EUC can go further into the environment of the system and also describe potentially relevant behaviour in the environment that does not happen at the interface. Moreover, an EUC does not necessarily fully specify which parts of the considered work processes will remain manual, which will be supported by the new system, and which will be entirely automated by the new system. An EUC can be similar to, in the sense of Lauesen [La03], a task description.

### AS EUC Tier 1: Analysis Documentation Produced by WM-data

At the workshops that we report on here, analysts from WM-data documented work processes (current and future), primarily by drawing swim lane diagrams as the one shown in Figure 2 which depicts the workflow that must be carried out, when a new user is created, who should be allowed to access the PUA network.<sup>3</sup> The input necessary to make the descriptions came from relevant personnel from PUA, who participated in the workshops.

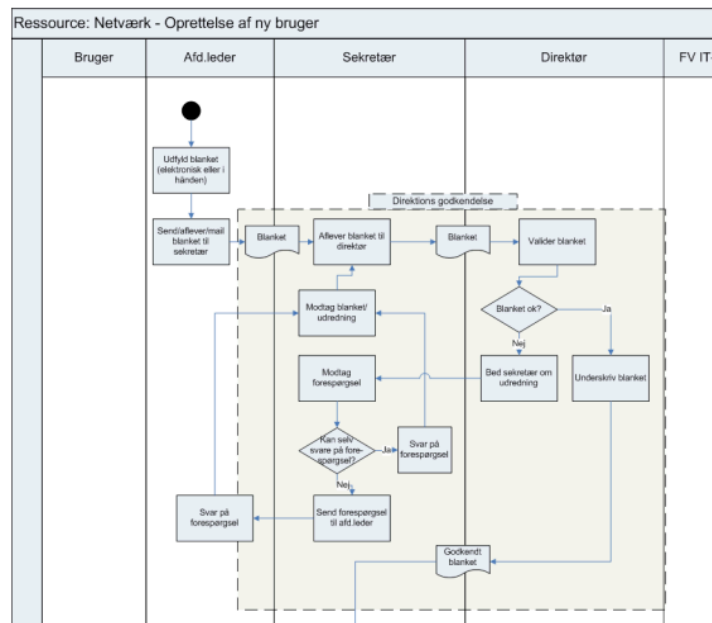


Figure 2 Partiel swim lane diagram showing creation of a new user of the PUA network

<sup>3</sup> The partial swimlane diagram is in danish and included to illustrate the diagrams that were used to create EUCs.

A number of swim lane diagrams were created. In total,  $3 \times 6 = 18$  different work processes are considered. The number 3 comes from the fact that, we consider three different action types, namely (1) creation, (2) update, and (3) closing of user accounts; the number 6 comes from the fact that we consider 6 different IT systems (or more generally, “resources”), namely the systems named Network, Navision, GIS, Xellent, EDoc, Geo Environ.

### AS EUC Tiers 2 and 3

Tier 2 is (1) a formalized version of tier 1, and (2) an execution engine that drives the graphical animation of tier 3. The animation tier, tier 3, is created with the help of Magee et al’s SceneBeans animation framework [MP00]; Figure 3 shows a snapshot.

The animation tier is consistent with the CPN model of the formal tier. At any time, the graphical animation represents the current state of the CPN model and mimics its execution. Technically, the link between the CPN model and the animation tier is that the CPN model calls drawing functions when it executes. The CPN model thus causes graphical objects like organisational unit icons and document icons to be created, moved, deleted, etc. in the animation.

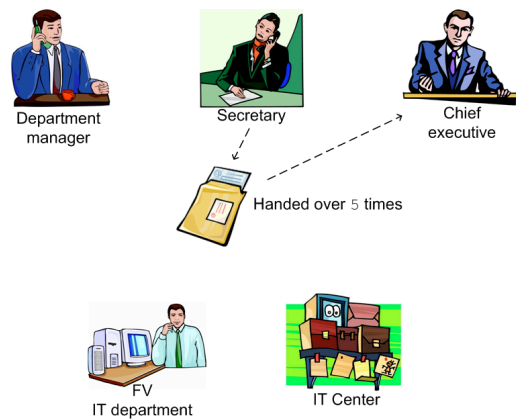


Figure 3 Snapshot of the animation tier of the EUC

Figure 3 mimics a situation, where five stakeholders are cooperating in the handling of a request to open a new user account. The stakeholders are the department manager, a secretary, the chief executive, the IT department of PUA, and an external IT centre. The document icon represents the request. In the current situation, the request is on its way from the secretary to the chief executive; the animation user sees the document icon moving. The piece of information attached to the icon counts how many times the modelled document has been handed over.

When the animation starts, it presents a dialog box that prompts the animation user to specify what he wants to mimic. Which system is in concern? Do we want to consider creation, update, or closing of a user account? It can also be specified whether that handling mode should be normal or urgent; in the latter case, some of the stakeholders which are involved in the normal handling are bypassed.

The animation layer is in general generated manually, as was the case of our EUC as well. This meant that we had to map each state change/transition in the model to a concept in the animation, so e.g. when the transition that moves a letter from one person to another occur in the model, we made the necessary SceneBeans command to reflect this in the animation.

### **Lessons learned**

Through meetings with PUA we learned that EUCs are valid in many phases in the lifecycle of the system that we are working towards. The lessons that we learned can be divided into four parts that we explain in the remainder of this section.

*EUCs offer better understanding of requirements.* Text documents as well as swim lane diagrams (Figure 2) were used to structure the requirements found in workshops by users and experts that knew of existing workflows. The attendees at the workshops found it is hard to get a full overview of the whole process just using text documents and swim lane diagrams, since these descriptions describe scenarios, rather than representing a complete model. Our EUC had all the requirements encoded as rules and when it is executed, it shows how all the requirements play together. A further improvement is that with the EUC, it is easy to get users to talk about requirements; since they did not have to talk of the requirements from e.g. the swim lane diagrams' box and arrows, instead they used terminology that the EUC's graphical presentation offer.

*EUCs are useful when selling a new system.* When presenting users for how a new system will work, we learned that text documents and swim lane diagrams, often seem inappropriate to convey the ideas behind the new design. The analysts at PUA thought that the EUC was a much better means of communication when talking to non-technical users or management that do not have time to fully understand the traditional requirement artefacts. Furthermore, PUA would like to present users of the system, how the current workflows are and how the future workflows will be, using the same graphical layer.

*EUCs give momentum in the Software Development Phases.* A software developer attended the meeting with PUA and software analysts when we presented the EUC. He claimed that this EUC gave him more contexts of the workflows that occur at PUA, with regards to who does what and when, and also that he thought that it was easier to remember them, than using e.g. swim lane diagrams.

*EUCs keep focus on what is important.* It was noted when we presented the EUC that in contrast to traditional prototyping, people actually discussed what was important at this stage of the analysis: The workflows. One person noted that over the years he has observed that if you show a person a screenshot of a user-interface and want him to discuss the workflow in the system, the person may begin to focus on the user-interface itself.

This could be because he does not have anything in particular to say about the workflow, but feels obligated to comment on something. In this way a lot of unnecessary and untimely discussions were generated. He felt that EUCs, because of their simple and non-user-interface appearance make people consider how the systems workflow are and should be, rather on e.g. prematurely designing the user-interface.

*EUCs are useful after deployment.* Artefacts generated in the analysis phase of a software project, are seldomly used for anything else but input to the design phase. One of the people at PUA thought that EUCs would be very useful as a supplement to manuals and other kinds of documentation of the system. For non-technical users these traditional ways of documenting the functionality of the system are hard to read and understand, so instead of using the documentation they will often resort to simply asking other people instead. The PUA person continued, and said that the EUC would help the individual to understand his or her role in the organization.

### **3 Experiment 2 - Activity Cases<sup>4</sup>**

#### **The experiment**

The experiment was carried out as a part of a pre-analysis project at a public Danish library, Vejle Library. The purpose of the pre-analysis project was to identify possible improvements of the library's mediation of library user's search for relevant information. A library user that requests search mediation will be serviced by a librarian. During a search session a librarian will use the library user's expressed information needs to search for relevant information via search engines and databases.

During the pre-analysis project a number of analysis and design activities were carried out. User simulations were used to establish understandings of the current activities. Modeling of current activities were used to capture aspects of these understandings. Formulation of future stories and brain storms were used to create visions about changed activities and new ways of using IT systems. Modeling of Future situations were used to capture aspects of the visions.

---

<sup>4</sup> Experiment 2 was carried out by the first author.

The purpose of the research project was to experiment with activity modeling by means of activity cases and interaction patterns. The research was carried out by a librarian and a researcher. The librarian was the primary actor in the pre-analysis project. The researcher served as a consultant in the pre-analysis project.

### Modeling techniques

We used activity cases, informal process models, and interaction patterns to model existing and envisioned information search activities.

An activity case defines selected characteristics of an existing activity or a vision about a new activity [Bæ05]. We use the following four aspects (NICE) to characterize activity cases. The *name* expresses the essence of the corresponding activity. The name is important because we use to refer to an activity and indicate what it is about. The *intention* expresses the purpose of an activity. The *content* of an activity can be described in terms of actions, events, actors, resources etc. The content can be described by means of, say, activity diagrams [RJ99], BPMN diagrams [Wh04], and EPC diagrams [De02], [LL06], or they can be described in terms of text. The *environment* of an activity can be described in terms of actors that interact or statically related with the activity.

We have used interaction patterns to define the structural aspects of the information search activities in the library. An interaction pattern defines a dynamic relation between two participants. At least one of the participants must be an actor [Bæ06].

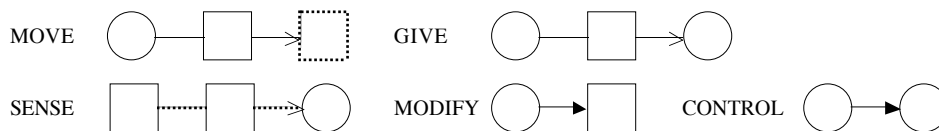


Figure 4 Interaction patterns

Figure 4 shows the notation we use to model interaction patterns. Circles represent actors. Rectangles represent objects (things or information). Dotted rectangles represent locations. MOVE is a pattern where an actor moves an object to a destination. GIVE is a pattern where an actor gives an object to another actor. SENSE is a pattern where an actor senses aspects of an object (the dotted line indicates that no visible, physical action is taken place). MODIFY is a pattern where an actor modifies an object. CONTROL is a pattern where one actor controls the activities of another actor.

### The situation

The experiment focused on an activity where a library user has a set of information. A librarian uses his understanding of these needs to search for information resources. The use and the library engage in a dialogue about the user's information needs, potential search terms, and the relevance of search results.



The information search activity is characterized by communicative interaction between user and librarian and it is characterized by a shared interaction with IT-systems like the Internet and library databases. Also, there is an important element of cognitive activities where the user and the librarian tries to understand the problem at hand and where they consider possibilities and think about formulations and search results.

### Activity case 1 – current situation

This activity case represents selected aspects of the current mediation of user’s information search. It is based on user simulations during which two librarian’s simulated the information search activity playing the roles of user and librarian.

**Name.** Information search.

**Intention.** To mediate a library user’s search for information resources.

**Content.** The leftmost diagram in Figure 5 represents the current search activity in terms of a set of activities that the librarian and the user carry out. The notation is informal. Rectangles represent objects and round-corner rectangles represent activities. *Formulate query* is an activity in which the user expresses information needs and the librarian asks questions and suggests interpretations. Based on this the librarian formulates a set of *search terms* that is used as input to a *search* activity in which the librarian uses the search terms to ask a query to a search system. The user and the librarian analyse and evaluate the *answer*—a set of objects that is returned from the search system (database, Internet search engine, etc.). *Select* is an activity in which the librarian uses “cut & paste” to copy relevant resources from an answer to the *resource collection*—a text document in which the selected resources from search answers are stored. *Finish* is an activity in which the resource collection is formatted and enhanced with clarifying comments.

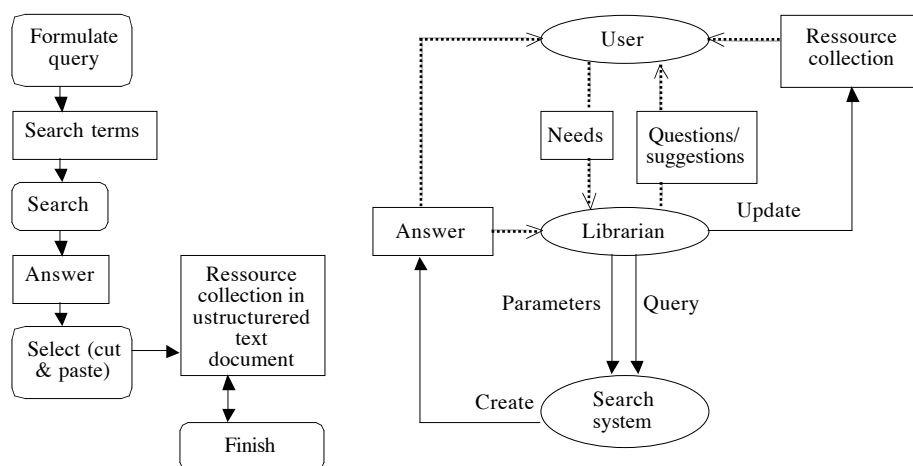


Figure 5 Current search activities

The rightmost diagram in Figure 5 represents the current search activity in terms of a set of interactions. Two persons, a librarian and a user, participates in the activities. The user expresses information needs. The librarian asks questions and suggests interpretations. The librarian adjusts search parameters and poses query with search terms to the search system. The search system creates an answer to a query. The librarian updates the resource collection that contains selected resources from query answers. The user can see the answers and the resource collection.

**Environment.** Other activities in which librarian and user engage.

### Activity case 2 – future situation

This activity case represents selected aspects of the current mediation of user’s information search. It is based on brain storms and analysis of Activity Case 1b.

**Name.** Information search.

**Intention.** To mediate a library user’s search for information resources.

**Content.** The leftmost diagram in Figure 6 represents the envisioned future search activity in terms of a set of activities that the librarian and the user carry out. The notation is informal. Rectangles represent objects and round-corner rectangles represent activities.

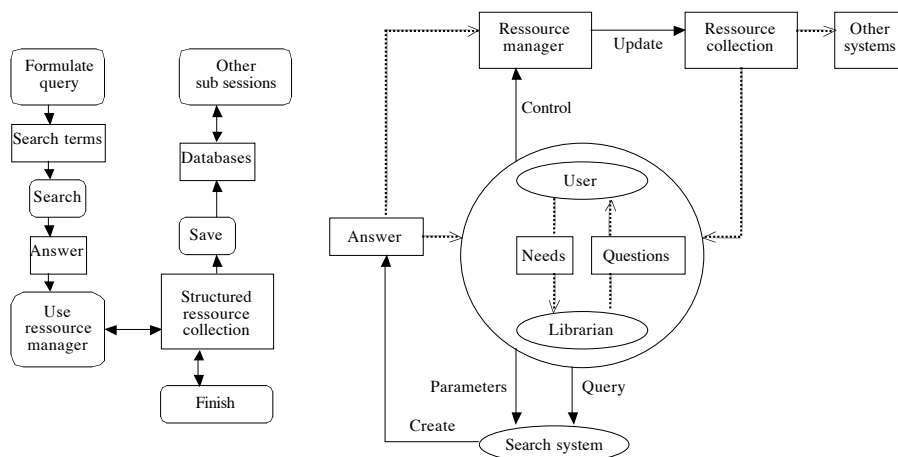


Figure 6 Future search activities

The major change when compared to the model in Figure 5 is that the model in Figure 6 presumes that an actor (IT system) called an Resource Manager can be used to manage the resources that are evaluated as relevant parts of the query answers. Rather than manually updating an unstructured resource collection then librarian uses the Resource Manager to select and modify selected resources.

The rightmost diagram in Figure 6 represents the envisioned future in terms of a set of interactions. Two persons, a librarian and a user, participate in the activities. The resource manager updates the resource collection. The user controls the resource manager. The user expresses information needs. The librarian asks questions and suggests interpretations. The user adjusts search system parameters. The user poses a query with search terms to the search system. The search system creates an answer to a query. The librarian adjusts search system parameters. The librarian poses a query with search terms to the search system. The librarian controls the resource manager. The resource collection is “read” by other systems.

**Environment.** Other activities in which librarian and user engage. Environment of other systems.

### **Lessons learned**

Activity case 1 were created in order to understand the existing situation and to support discussions about potential improvements. Activity case 2 represents the results of these discussions. The following lessons are based on the observed experiences from the use of activity cases and from informal conversations with library staff.

*Activity cases support modeling flexibility.* Activity cases support a balanced focus on the elements of a modeled activity that are considered relevant, i.e., actors, activities, things, and information etc. No specific modeling languages are presumed. Any combination of formal and informal modeling languages can be used. This turned out to be very relevant in the library case because the informal activity flow modeling notation turned out to give an insufficient view of the activities. The library case is characterized by many iterative interactions that are not easily captured in terms of activity flow. The interaction patterns turned out to supplement the activity flows.

*Activity cases support innovation.* Interaction patterns facilitate creative discussions about the roles played by IT-systems in business activities. The use of interaction patterns in activity cases turned out to be useful in creative discussions about potential uses of IT-systems to improve the library search activities. The reason is that all interactions can be mediated by actors that are added between the interacting elements. For example, the Resource Manager can be viewed as a mediator that mediates the librarian’s interaction with the selected resources.

*Activity cases go beyond workflow modeling.* An activity case focuses on a selected activity system in terms of its name, intention, content, and environment. This implies that the modeler is encouraged to go beyond mere workflow modeling. In the library case we modeled the selected activity system (information search) in terms of both workflow and interaction patterns because the workflow perspective turned out to be an inadequate representation of the roles of mediating tools in the search process.

## 4 Discussion

The following scenarios describe potential combinations of EUCs and activity cases. They are based on the lessons learned from the two experiments. All scenarios are based on the assumption that the requirements engineering process is structured around three EUC-tiers.

*Scenario 1 – Activity cases may be used as organizing principle.* When the three EUC-tiers are created a significant amount of models may be created. These models can be organized as activity cases to ensure that all models are named in a coherent manner and contextualized in terms of their intentions and environments. The NICE template offers a framework for model organization. Different models of a specific activity can be traced if all models (Tier 1, Tier 2, Tier 3) are contextualized by means of the NICE template and if a coherent naming scheme is used.

*Scenario 2 – Activity cases may be used to support innovation.* Activity cases can be used to support innovation and creativity when Tier 1 is created. Activity Cases are useful for gaining the first ground in understanding a problem, and they do this by structuring the domain in actors, resources, actions, and events. EUCs on the other hand are useful making a full behavioral model, of the discrete descriptions captured by each activity case, and to establish in a greater degree the resource perspective and control-flow perspective of the overall system. EUCs are used to support deep understanding of the dynamics of complex processes.

*Scenario 3 – Activity cases may be used to support flexible clarification.* Activity cases are used to support fast experimentation with processes that have turned out to be problematic in EUCs. When a process model is animated one or more deficiencies may be detected. In such situations the flexibility of activity cases may be used to experiment with ways to overcome each identified deficiency without having to formally model the process.

The next series of evaluation experiments in our design science project will be based on scenarios like these. The experiments will be based on industrial requirements engineering activities in which a set of promising scenarios will be evaluated. The scenarios implicitly characterize a requirements engineering process that combines flexibility and agility with formality and animated processes. It is very likely that such a process could have been used to improve our to experiments. The public services case could have used the NICE template to organize the various models and it could have benefited from the flexibility offered by Activity Cases. The library search case could have benefited from the animations offered by EUC. Animations of the search process could be distributed to relevant librarians and selected user as a basis of a more thorough validation of the proposed new search activity.

## 5 Conclusion

We have presented two different requirements engineering techniques called EUCs and Activity Cases. An EUC consists of three different models of the work processes that must be supported by a new system. Tier 1 is an informal model. Tier 2 is a formal, executable model that is based on Tier 1. Tier 3 is a graphical animation of tier 2. Tier 1 is created in an informal modeling activity. Tier 2 is created in a formal modeling activity. Tier 3 is an animation activity in which the formal model from Tier 2 is animated. An activity case is model of a work activity. It is based on a templete with four hedalines: Name, Intention, Content, and Environment. Activity cases can be used to experiment with models of existing and envisioned business processes. For each technique we have described and discussed an experiment in which the technique has been applied and evaluated. And we have sketched three scenarios that represent possible combinations of EUCs and Activity Cases. Future research includes industrial experiments that are based on these scenarios. The purpose is to identify characteristics of a requirements engineering process that combines flexible modeling, formal modeling, and process animation.

## References

- [AW90] Avison, D. E. and A. T. Wood-Harper (1990). Multiview, Blackwell Scientific Publications.
- [BH98] Beyer, H. and K. Holtzblatt (1998). Contextual Design. Designing Customer-Centered Systems, Morgan Kaufmann.
- [BJ04] Bossen, C. and J. B. Jørgensen (2004). Context-Descriptive Prototypes and their Application to Medicine Administration. DIS'2004. Designing Interactive Systems. Cambridge, Massachusetts, Acm Press.
- [Bæ05] Bækgaard, L. (2005). From Use Cases to Activity Cases. ALOIS'05. Action in Language, Organisation and Information Systems. Limerick, Ireland.
- [Bæ06] Bækgaard, L. (2006). Interaction in Information Systems - Beyond Human Computer Interaction. ALOIS'06. Action in Language, Organisation and Information Systems. Borås, Sweden.
- [Co01] Cockburn, A. (2001). Writing Effective Use Cases, Addison-Wesley.
- [De78] De Marco, T. (1978). Structured Analysis and System Specification. Yourdon., Yourdon.
- [De02] Dehnert, J. (2002). Making EPCs fit for Workflow Management. EPK'2002 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. Trier, Germany.

- [HM04] Hevner, A. R., S. T. March, et al. (2004). "Design Science in Information Systems Research." MIS Quarterly **28**(1): 75-105.
- [Ja01] Jackson, M. (2001). Problem Frames - Analyzing and Structuring Software Development Problems, Addison-Wesley.
- [JB04a] Jørgensen, J. B. and C. Bossen (2004). Executable Use Cases as Links Between Application Domain Requirements and Machine Specifications. 3rd International Workshop on Scenarios and State Machines (at ICSE'2004). Edinburgh, Scotland, IEEE.
- [JB04b] Jørgensen, J. B. and C. Bossen (2004). "Executable Use Cases: Requirements for a Pervasive Health Care System." IEEE Software **21**(2): 34-41.
- [La03] Lauesen, S. (2003). "Task Descriptions as Functional Requirements." IEEE Software **20**(2): 58-65.
- [LL06] Lübke, D., T. Lücke, et al. (2006). Using Event-Driven Process Chains for Model-Driven Development of Business Applications. Workshop XML4BPM 2006 - XML Integration and Transformation for Business Process Management. Passau, Germany.
- [MP00] Magee, J., N. Pryce, et al. (2000). Graphical Animation of Behavior Models. 22nd International Conference on Software Engineering. Limerick, Ireland, ACM Press.
- [MS95] March, A. T. and G. F. Smith (1995). "Design and Natural Science Research on Information Technology." Decision Support Systems **15**: 251-266.
- [RJ99] Rumbaugh, J., I. Jacobson, et al. (1999). The Unified Modeling Language Reference Manual, Addison-Wesley.
- [Wh04] White, S. A. (2004). Introduction to BPMN, WWW.BPMN.ORG.

# UML 2 Profiles for Ontology Charts and Diplans

## Issues on Metamodelling

Jose Cordeiro<sup>1</sup> and Kecheng Liu<sup>2</sup>

<sup>1</sup>EST Setúbal/IPS, Rua do Vale de Chaves, Estefanilha,  
2910-761 Setúbal, Portugal  
[j.cordeiro@computer.org](mailto:j.cordeiro@computer.org)

<sup>2</sup>The University of Reading, Whiteknights,  
Reading, RG6 6AF, UK  
[k.liu@reading.ac.uk](mailto:k.liu@reading.ac.uk)

**Abstract.** Organisational Semiotics (OS) uses Ontology Charts (OC) for requirements representation. This technique that shows affordances and their ontological dependencies constitutes the essential diagrammatic communication facility of this theory. On the other hand Diplans diagrams are in a similar way the main mean of expression of the Theory of Organized Activity (TOA). Diplans show us bodies and (human) actions and their relationships. Both theories belong to the socio-technical perspective of information systems development and were chosen as part of a unification work that includes both. Regarding UML, it is a *de facto* standard and it is seen as a powerful and widely accepted technique for modelling. To represent OC and Diplans with UML will most benefit the underlying theories by widening their audience and enabling to use the numerous available tools.

This paper proposes two new UML 2 profiles for representing respectively, OCs and Diplans. Examples of application of both profiles are shown and an extended discussion on their creation is made. Our concern is to bring to discussion the different issues that came forward when metamodelling both solutions and, consequently, to assess the feasibility of UML for this purpose.

## 1. Introduction

Modelling plays a major role in the way we perceive, plan and act within a particular context. Models show us *simplifications of the reality*, usually by representing and emphasizing some key elements of this reality. Each theory defines its own reality or context and has its particular view of the world. The major elements of these theories are commonly shown in models, thus highlighting their key concepts. In order to represent these central concepts many theories developed a particular diagrammatic language where the concepts and their relationships are shown. This is the case of Organisational Semiotics (OS) which uses Ontology Charts (OC) for modelling concepts such as *affordances* and *ontological dependencies*. A second case, which is used in this paper, is Diplan that is another diagrammatic language applied by the Theory of Organized Activity (TOA), to express the concepts of *bodies* and (*human*)

*actions* and their relationships. These theories were chosen as part of a unifying work that intends to merge them both. Important gains are obtained by using UML and in particular UML Profiles instead of those particular kinds of diagrams, examples are:

- A much wider audience will be able to understand and use the diagrams
- A uniform and less ambiguous (formal) way of representing the elements and their relationships
- The possibility to use the numerous available UML tools
- A possible and easy way to exchange and to include diagrams in applications as part of the requirements and/or documentation

In this sense this paper presents and proposes two new UML 2 profiles for representing, respectively OCs and Diplans. Besides the creation of these profiles the creation process itself happens to be a great challenge and many issues were raised. So, we intend to report here as well the difficulties and the issues that emerged from the creation process. We think that many of the problems found will be common to other researchers, that our solutions will be helpful to them and the issues raised will contribute to the discussion and to the enhancement and evolution of UML in general and UML metamodelling in particular.

This paper is organised as follows: section 2 will present the related work, section 3 will be devoted to the necessary theoretical background on TOA and OS, UML proposed profiles will be shown and exemplified in section 4, section 5 will present the discussion and rationale for the created profiles and finally, conclusions will be given in section 6.

## 2. Related Work

We found a minor number of attempts to use UML within OS. In [LO99] an UML activity diagram is extended in order to support *norm* specifications. *Norms* are not depicted in OCs but are directly attached to each individual affordance shown in these charts. OCs doesn't provide any means to represent these norms and this work is useful as an extension of OCs. Also [SD03] presented another work related to *norms*. In this case *use cases* are derived from *norm analysis*. There is a small part of their work associated with this particular derivation and not much relevant.

A recent paper and the most significant for the use of UML in OS is the paper of [Bo04] where some heuristic rules for class diagram derivation from OCs are proposed. Even so, this work just gives some 'simple' hints on how to obtain (and translate) the OC elements into UML elements. Used UML elements were limited to classes and associations, compositions and generalizations relationships among them.

To the best of our knowledge no other research work tried to apply UML profiles to produce specialized UML diagrams for expressing OCs. The same is true about Diplans. In fact there was no other related work which relates UML and TOA.

Regarding UML profiles much work has been done and we will point some references afterwards when appropriate.



### 3. Ontology Charts and Diplans Theoretical Background

#### 3.1. Organisational Semiotics and Ontology Charts

OS applies Semiotics to the study of business systems and organisations. Within OS the work of Ronald Stamper is the most advanced and influential and is the one usually referred as OS in general while denoting Stamper’s particular theory and view (see for example [St73], [St96], [St00] and [Li00]). OC is the diagrammatic language used in Stamper’s OS and is the outcome of applying the method of Semantic Analysis (SAM) to an organisational problem. OCs are mainly used for organisational requirements offering a precise and stable view. *Affordances* and *Ontological dependencies* are the key elements depicted in these charts. *Affordances* are the invariants of the environment and represent patterns of behaviour afforded by some agent. As an example, a cup may be considered an affordance because it affords drinking, holding liquids, throwing it, and other actions (or patterns of behaviour). *Ontological dependencies* (OD) are existential dependencies between affordances where some affordances cannot exist without the existence of others. For example *swimming* is not possible without being *immersed in water*. The *swimming* affordance requires the *existence* of both a water affordance and an immersed agent affordance. Agents and affordances are represented as nodes in OCs, while ODs are the lines connecting these nodes. Agents and affordances allow specific/generic relationships between them. Also ontological dependencies can model existential relationships between a whole and a part. Affordances can be substantive, representing here-and-now or semiological, standing for other affordances. On the other hand agents can have roles in the scope of an OD. Affordances can also be Universal or Particular corresponding to a concept similar to respectively type and instances. Time is also present in OCs – leftmost affordances must exist before affordances on the right side to exist. Last elements represented in OCs are determiners which generalize the

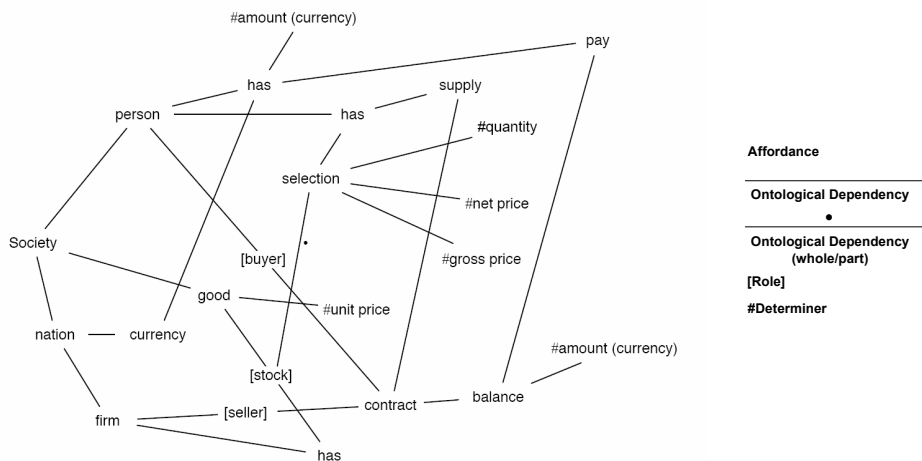


Fig. 1. An Ontology Chart of a grocery shop (proposed by Ronald Stamper)



## 4. UML Profiles for Ontology Charts and Diplans

### 4.1. The OS UML Profile

The UML profile metamodel created for OC is depicted in figure 3. The stereotypes and constraints defined for this profile are detailed in table 1. Besides these UML extensions a particular diagram – the OC Diagram – was created in order to represent a standard OC with UML. Discussion of the creation of this profile is made in the next section.

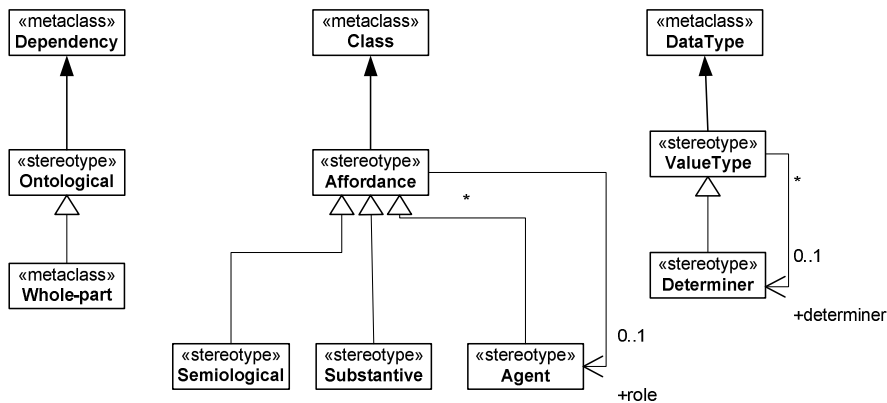

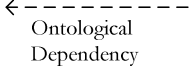
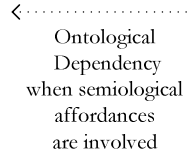



Fig. 3. OS Profile metamodel

Table 1. OS profile stereotype definitions

<b>Name</b>	<b>Affordance</b>	
<b>Extended Class</b>	Class	
<b>Description</b>	Represents the Affordance concept. An affordance is an element that enables a group of actions for an agent.	
<b>Constraints</b>	An affordance cannot have attributes. Affordances operations are abstract.	
<b>Notes</b>	Affordance operations can be used to specify operations that are enabled by the affordance	
<b>Name</b>	<b>Agent</b>	
<b>Base Class</b>	Affordance	<b>Notation</b>
<b>Description</b>	Describes an agent Affordance. An agent represents a person or a legal entity	 Optionally an agent can be shown inside an oval figure
<b>Constraints</b>	-----	

<b>Name</b>	<b>Semiological</b>	
<b>Base Class</b>	Affordance	
<b>Description</b>	A special kind of affordance representing a speech act.	
<b>Constraints</b>	-----	
<b>Notes</b>	Also described as a semiotic sign: something that stands for another thing, not the actual thing.	
<b>Name</b>	<b>Substantive</b>	
<b>Base Class</b>	Affordance	
<b>Description</b>	Represents the common affordance, which is different from an Agent or a semiological affordance	
<b>Constraints</b>	-----	
<b>Name</b>	<b>Ontological</b>	
<b>Extended Class</b>	Dependency	<b>Notation</b>
<b>Description</b>	An Ontological relationship is a binary dependency between two affordances where the dependent affordance cannot exist without the existence of the other.	 
<b>Constrains</b>	1) This relationship can only be applied between affordances. 2) It is a binary relationship. 3) An affordance can only be the target of at most two Ontological dependency relationships.	
<b>Notes</b>	A dotted line may be used for dependencies on semiological affordances.	
<b>Name</b>	<b>Whole-part</b>	
<b>Base Class</b>	Ontological	<b>Notation</b>
<b>Description</b>	An Ontological whole-part relationship is a binary dependency between two affordances where the dependent affordance represents the part that cannot exist without the whole.	
<b>Constrains</b>	-----	
<b>Notes</b>		
<b>Name</b>	<b>ValueType</b>	
<b>Extended Class</b>	DataType	
<b>Description</b>	Used with affordances to express information about properties and/or parameters.	
<b>Attributes</b>	determiner: ValueType [0..1] A kind of quantity that is identified by an instance of the determiner stereotype.	
<b>Constraints</b>	The determiner attribute must reference a ValueType to which the «determiner» stereotype has been applied.	
<b>Notes</b>	Stereotype imported and adapted from SysML profile (OMG, 2007c)	
<b>Name</b>	<b>Determiner</b>	
<b>Base Class</b>	ValueType	
<b>Description</b>	A kind of quantity that represents a measurement dimension. For example size, height, volume, identifier.	
<b>Constraints</b>	The “determiner” attribute inherited from the ValueType stereotype must not contain any value.	
<b>Notes</b>	Stereotype imported and adapted from SysML profile (OMG, 2007c)	

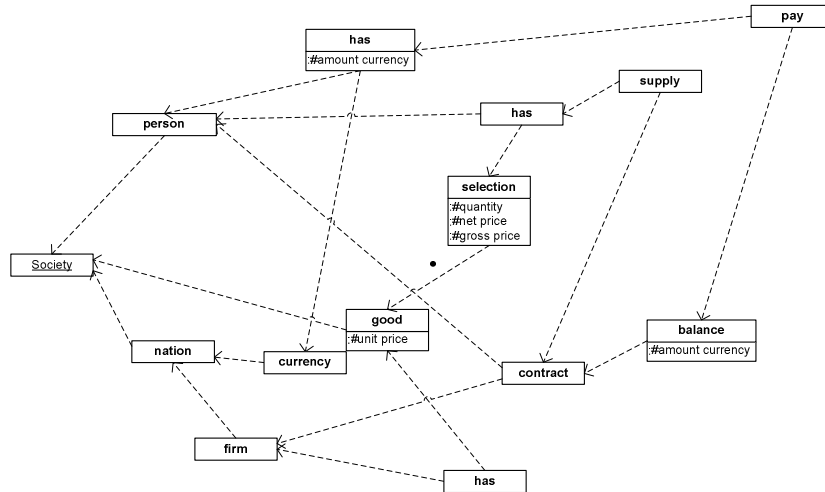


Fig. 4. An OC diagram using the OS Profile

The OC Diagram defined in this profile is a special case of a UML Class diagram. This diagram is used for showing affordances and their ontological dependencies. All OC diagrams are provided with an immutable instance specification of an *agent affordance* named *Society*. This instance should be the root of all ontological dependencies in the diagram where ultimately all affordances are ontologically dependent on. Because the only kind of dependency shown in OC diagrams is the ontological dependency it will be possible (and optional) to omit the stereotype keyword. Another notation rule is to respect the OC rule that states that all dependencies must be depicted from left to right. This means that affordances dependent on other affordances should appear at right of the affordances from which they depend. If the UML tool adopts and verifies this criterion within OC diagrams then it will be possible to hide all direction information from the dependency lines. In figure 4 the example given in figure 1 is reproduced with the OS profile applied to it.

#### 4.2. The TOA Profile

The Diplan metamodel is given in figure 5 and the description of all created stereotypes and constraints is presented in table 2. Discussion of this profile will be given in the next section as well.

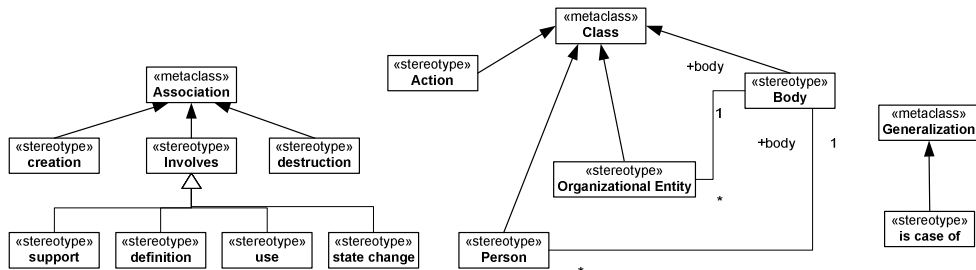
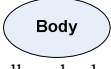

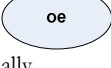
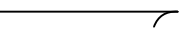
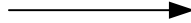

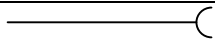
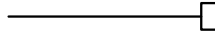
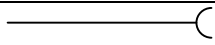



Fig. 5. TOA Profile metamodel.

Table 2. TOA Profile stereotype definitions

Name	Action	
Extended Class	Class	
Description	Represents an (human) action concept	
Constraints	-----	
Name	Body	
Extended Class	Class	Notation
Description	Represents the TOA concept of a body. A body is a <i>material</i> element	 Optionally a body can be shown inside an oval figure
Constraints	-----	
Notes	A body can and usually have states that can be shown in the usual UML way	
Name	Person	
Extended Class	Class	Notation
Description	Represents the TOA concept of a person.	 Optionally a person can be shown inside an oval figure
Attributes	body:Body Represents the body of the corresponding person as a material element	
Constraints	-----	
Notes	Usually the body is not shown.	
Name	OrganizationalEntity	
Base Class	Class	Notation
Description	Represents the TOA concept of an Organizational Entity	 Optionally an organizational entity is inside an oval figure
Attributes	body:Body Represents the body of the corresponding organizational entity	
Constraints	-----	
Notes	Usually the body is not shown.	
Name	IsCaseOf	
Extended Class	Generalization	Notation
Description	Is case of has a similar meaning to the UML generalization element	
Constraints	The general and specific classifiers should be both bodies or both actions	
Notes	The notation is the same as for Diplan and the semi arrow points to the generic element	

<b>Name</b>	<b>Involves</b>	
<b>Extended Class</b>	Association	
<b>Description</b>	An involvement relationship relates an action with a body	
<b>Constraints</b>	1) This relationship must be applied between an action and a body. 2) It is a binary relationship	
<b>Notes</b>		
<b>Name</b>	<b>creation</b>	
<b>Extended Class</b>	Association	<b>Notation</b>
<b>Description</b>	Describes the creation of a body by an action	
<b>Constraints</b>	Navigation is in the direction of the action to the body	
<b>Notes</b>	A solid arrow is used for the association end	
<b>Name</b>	<b>destruction</b>	
<b>Extended Class</b>	Association	<b>Notation</b>
<b>Description</b>	Describes the destruction of a body by an action	
<b>Constraints</b>	Navigation is in the direction of the body to the action	
<b>Notes</b>	A solid arrow is used for the association end	
<b>Name</b>	<b>support</b>	
<b>Base Class</b>	Involves	<b>Notation</b>
<b>Description</b>	Support is effort towards the continued existence and/or improvement of the body	
<b>Constraints</b>	Navigation is in the direction of the action to the body	
<b>Notes</b>	Semi circle end is used for the navigation end	
<b>Name</b>	<b>definition</b>	
<b>Base Class</b>	Involves	<b>Notation</b>
<b>Description</b>	When a body is in a specified state by <i>definition</i>	
<b>Constraints</b>	Navigation is in the action – body direction	
<b>Notes</b>	Semi square end is used for the navigation end	
<b>Name</b>	<b>Use</b>	
<b>Base Class</b>	Involves	<b>Notation</b>
<b>Description</b>	When a body is <i>used</i> by an action	
<b>Constraints</b>	Navigation is drawn in the direction of the body to the action	
<b>Notes</b>	Semi circle end is used for the navigation end	
<b>Name</b>	<b>stateChange</b>	
<b>Base Class</b>	Involves	<b>Notation</b>
<b>Description</b>	Means that a body changes its state because of the associated action	
<b>Constraints</b>	-----	
<b>Notes</b>	A stereotype is used for this representation	

As in the case of the OS profile a new kind of diagram is defined for use with the TOA profile - the Diplan diagram. This diagram is used to show action, bodies and their involvement relationships. The *involves* stereotype may be omitted because all the association relationships depicted in the Diplan diagrams are of this kind. As before the example given in figure 2 is reproduced in figure 6 with the TOA profile applied to it.

## 5. Discussion

One of the key reasons to choose UML profiles in this work instead of defining a complete metamodel for OCs and Diplans representation is the possibility to use UML tools. Among other benefits these tools allow for model interchange, model validation and model storage. In this sense UML profile construction becomes an unavoidable and mandatory process. In order to build these profiles some guidelines should be followed. (see for example [FV04]). In a general and simple view these guidelines recommend us to create a specific domain metamodel, to choose from this metamodel the relevant elements, to extend the appropriate UML metamodel elements with some of these elements and to define additional constraints and tagged values (see [Ru05]). Although simple, this process has many issues, difficulties and compromises when we are metamodeling non object-oriented theories. In the next sub-sections some of the problems felt will be exposed and discussed. It should be stated as well that both UML profiles were created using the version 2.1.1 of the UML superstructure and infrastructure [Om07a], [Om07b].

### 5.1. The OS profile

Following the guidelines for profile creation drive us to an initial step, which is to find corresponding UML elements for OC elements. This is straightforward in this case, the ontological dependency clearly maps to a kind of dependency and the extension of the UML dependency element become obvious. Regarding affordances no similar concept exists within UML and the common solution in these cases is to extend the metaclass “class”. In fact this extension is very well adapted in this case because it allows the use of the ‘generalization’ relationship for affordances, a notion that is already present with affordances. Another benefit is the possibility to express normal affordances as classes corresponding to *Universals* in OS and specific affordances as instance specifications corresponding to *Particulars*. Generalization can also be used in the metamodel for distinguish the different types of affordances, namely agents, substantive and semiological affordances.

A first problem appears because we need to express roles in dependency

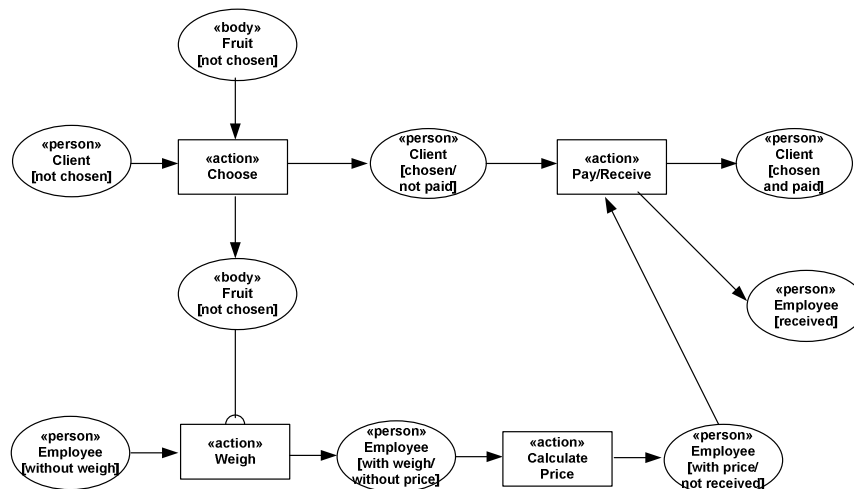


Fig. 6. A Diplan diagram using the TOA Profile



relationships. The dependency relationship doesn't allow roles to be associated with the target and/or source elements. Alternatively, it would be possible to extend the UML association for the ontological dependency in order to express roles but this solution would lose its expressive power. In fact, the OS profile was made considering its intended users, so it was important that obtained models were close to the UML notation (and semantics). The objective was to increase the number of people that would understand OC diagrams. Therefore the goal was to be close to the standard UML semantics and notation and to avoid new notations in this profile.

A second difficulty is: how to express the *determiner* concept? A determiner is a generalization of a measurement and it is an affordance as well. In this case it was modelled as an attribute associated with an affordance. As attributes, determiners must be *ontologically dependent* on their classes, thus this relationship becomes implicit and doesn't need to be shown. This solution also obliges old OC creators to understand this new notation.

Concerning the representation of the different types of affordances the solution makes necessary to use stereotypes to identify each kind. Visually this solution allows users to better understand the affordances used in a particular model. As been said before no new notation was given in this case.

A last problem occurs with an interesting rule that is followed when creating traditional OCs: affordances that depend on other affordances should be on the right of the affordances on which they depend. Time is introduced according to this rule, all left affordances exists before right affordances exist. UML has a poor representation of time and doesn't provide any spatial information about the elements. Except for a few cases, it is possible to put an UML element in any place in the model area and not much concern is made about its position. The solution for tool creators is to consider this new information in OC diagrams in order to make the rule effective. One more rule that cannot be formalized in UML is that each OC diagram must have an affordance instance specification named *Society* which is the root of ontological dependencies. These considerations lead to an issue in UML related to diagrams. A UML diagram is (surprisingly) not seen as a model element and it is created outside the metamodel. The new OC diagram and its elements is just a group of recommendations and rules in its application and no formal criteria are adopted for its definition.

## 5.2. The TOA profile

The TOA profile adopts a different approach from the OS profile; it is oriented towards a close connection to its original notation. Although the resulting models are UML models, the appearance will resemble the original Diplan diagrams. Regarding the main elements represented in Diplans, namely *bodies*, *actions*, *persons* and their *involvement* relationships, their *translation* to associated UML profile elements is not so straightforward. Bodies are material elements and no similar concept exists in the UML metamodel except for artifacts but this concept is too much focused on deployment and software elements and not suited for our goals. Therefore the usual class extension is the natural solution in this case. As a result *involvement* is also naturally expressed as an association extension. The main issue is about the *action* concept. There is an *action* element but it is not a classifier and it cannot be used with

an association according to the UML metamodel specification. This limitation doesn't allow us to express non directed relationships between actions and classes. Even in object oriented programming it is possible to classify operations (another kind of actions) and relate them with classes (for example actions can be modifiers, selectors, etc.) but the expression of these notions is not allowed in a simple way. An additional (and surprisingly as well) characteristic in the UML metamodel is the absence of a simple *relationship* metaclass between elements; this relationship is available for directed relationships through the dependency relationship but not for non directed relationships. The only relationship of this type is the association relationship but it is special because it is a classifier as well, and should relate two classifiers. This becomes a problem for the representation of the *person* concept. Naturally this concept could be expressed using the *actor* element (which is a classifier) but, in this case, there are limitations as well given the scope of the association relationship. An actor cannot be simply associated with a class because it doesn't have properties (as a class does) and therefore cannot own an association end. In this case no roles can be connected with the actor. Any association between an actor and a class must own both association ends, thus navigation is not possible. This restricts the use of the different types of involvement. So, the solution was to turn the *person* concept into a stereotype using a UML class extension.

Concerning the different types of involvement they are represented using stereotypes deriving ultimately from the UML association element and keeping the original notation. Let us just add a note for the person and organizational entity elements, which possess a body that it is represented as an attribute in the corresponding stereotype. Also a body is the only element that can have states according to TOA, this adapts perfectly to the body being a class.

A last difficulty found and not solved in the TOA profile was how to express a multiple relationship existing in the *state change* involvement. In the original diagram it was possible to dissociate the states from a person's body by connecting the previous and next states linked to the involvement relationship in the middle of it. This kind of graphic element is not available in UML and it is not syntactically and semantically possible to build a similar construct from the available UML elements.

### 5.3. General Remarks

Different strategies were identified when metamodelling both profiles regarding the selection of the UML metamodel elements to extend. In the OS profile the approach was to create a notation close to the traditional UML notation which leads to a selection of elements whose semantics was close to common UML. The TOA profile adopted a different criterion: the notation would have to be close to the original Diplan notation. We think that a third approach that would favour the similarity (semantics) between both the UML elements and the elements from the considered theory could also be possible.

Regarding the UML metamodel we found many problems when metamodelling both profiles. The UML metamodel is designed according to an Object Oriented approach and it shows many limitation when used to express different domains and approaches. As an example the limitation found to relate diagrammatically some UML elements such as classes and actors or classes and actions can be extrapolated to

other elements. Also, it lacks a normal relationship between model elements. Another problem not mentioned before is that the UML metamodel mixes different concerns in its hierarchy whereas other elements are not even considered. For instance some metaclasses refer to model elements aspects such as `PackageableElement`, `NamedElement` that are mixed with other aspects while elements such as diagrams, positioning aspects are not present.

**Table 3** - Summary of identified UML issues

UML issue	Comments
Notation may vary	It was possible to adapt both OS and Diplan profiles to a notation that was similar to, respectively, standard UML or to the original diagrammatic notation. This is a problem for users, making difficult to understand different models when applying relatively different notations.
UML metamodel elements usually have hidden aspects	Some simple UML elements like <i>action</i> , <i>actor</i> , <i>state</i> and others cannot be used to represent similar concepts because they cannot be freely associated with other elements. This is hidden and it is a consequence of the rigidity of the UML metamodel when defining these elements.
Relationships between elements are limited	The UML metamodel doesn't have a concrete relationship metaclass between elements, the usual solution is to use the association relationship that obliges the concepts to be represented as classes
UML metamodel elements with limited combinations among them	The UML metamodel limits our capability to combine different metamodel elements. A simple example was the impossibility to use roles with dependency relationships.
A diagram is not an UML metamodel element	It is not possible to adequately formalize relationships between diagrams and model elements. For example spatial information about model elements in a diagram cannot be used. Also number limitations of model elements in a diagram can't also be expressed. There are more examples of this problem.

## 6. Conclusions and future work

In this paper two UML profiles for Ontology Charts and Diplans were introduced. These profiles will permit the underlying theories, respectively Organisational Semiotics and the Theory of Organized Activity to use UML tools with several benefits as follows:

- Possibility to communicate the diagrams to software development teams and to include them with other diagrams in the same software project
- Interoperability of the diagrams with other model tools
- Consistency and verifiability of the diagrams
- Formalization of the diagrams

Besides these benefits OS in particular will gain with notation normalization. In fact, as we can observe from different works (for example [St96], [Li00]) different notations for OCs have been used. When using the OS profile, the associated UML tool will provide a single notation which should be used by all users, thus leading to a standard notation.

Concerning profiles creation this paper has raised different issues, from UML deficiencies and missing components to different strategies for profile development. Table 3 summarizes most of the issues found in this process.

This work was part of a research project that has as a goal to create a unified and fundamental theory for software development that will merge some relevant concepts of both OS and TOA. Future work will reuse created profiles for the modelling of this new theory.

## References

- [Bo04] Bonacin, R., Baranauskas, M. C. C. and Liu, K. 2004. From Ontology Charts to Class Diagrams - Semantic Analysis Aiding Systems Design. *In Proceedings of the 6th International Conference on Enterprise Information Systems, ICEIS 2004*, Porto, Portugal. v. 1. p. 389-395
- [FV04] Fuentes, L. and Vallecillo, A., 2004. An Introduction to UML Profiles. *UPGRADE, The European Journal for the Informatics Professional*, 5(2):5-13, April 2004. ISSN: 1684-5285
- [Ho97] Holt, A. 1997, *Organized Activity and Its Support by Computer*, Kluwer Academic Publishers, Dordrecht, The Netherlands.
- [Ho88] Holt, Anatol W., 1988, 'Diplans: a new language for the study and implementation of coordination'. In *ACM Transactions on Information Systems (TOIS) Volume 6, Issue 2*, pages: 109 – 125.~
- [LO99] Liu, Kecheng and Ong, Tina (1999) A Modelling Approach for Handling Business Rules and Exceptions, *The Computer Journal*, 42(3), 221-231.
- [Li00] Liu, K. 2000, *Semiotics in Information Systems Engineering*, Cambridge University Press, Cambridge, UK
- [Om07a] OMG, 2007a. Unified Modeling Language Superstructure Specification, v2.1.1. Available: <http://www.omg.org/cgi-bin/doc?formal/07-02-05> (May 2007)
- [Om07b] OMG, 2007b. Unified Modeling Language Infrastructure Specification, v2.1.1. Available: <http://www.omg.org/cgi-bin/doc?formal/07-02-06> (May 2007)
- [Om07c] OMG, 2007c. SysML FTF convenience document. Available: <http://www.omg.org/cgi-bin/doc?ptc/2007-02-04> (May 2007)
- [Ru05] Rumbaugh, J., Jacobson, I. and Booch, G., 2005. *The Unified Modeling Language Reference Manual (2nd edition)*, Addison-Wesley, Reading, MA
- [SD03] Shishkov, B. and Dietz, J. 2003, Deriving Use cases from Business processes, the Advantages of DEMO. *In: Enterprise Information Systems V*. Eds. O. Camp, J.B.L. Filipe, S. Hammoudi, and M. Piattini. Kluwer Academic Publishers, Dordrecht/Boston/London, 2004.
- [St96] Stamper, R. 1996, 'Signs, Norms, and Information Systems', in *Signs of Work*, eds. Holmqvist B. et al, Walter de Gruyter, Berlin.
- [St73] Stamper, R., 1973, *Information in Business and Administrative Systems*, John Wiley and Sons, Inc., New York.
- [St00] Stamper, R., 2000, Information Systems as a Social Science: An alternative to the FRISCO Formalism, in Falkenberg, E., Lyytinen, K. and Verrijn-Stuart (Eds), *Information System Concepts: An Integrated Discipline Emerging*, Kluwer Academic Publishers, Massachusetts, pages: 1-51.

# Service Identification and Design – A Hybrid Approach In Decomposed Financial Value Chains

Falk Kohlmann

Information Systems Institute  
University of Leipzig  
Marschnerstraße 31, 04109 Leipzig, Germany  
kohlmann@wifa.uni-leipzig.de

**Abstract:** Service-orientation is recognized as an important enabler for increasing efficiency and flexibility of transformation processes in business. Based upon the necessity of meeting dynamic customer needs and supporting organization concepts with numerous partners within emerging networks, flexible bundling of business processes is a key requirement. Service models derived from business and shared within a network can foster this flexibility. However, there is a lack of methodologies for combining technical-driven and business-driven service identification and clustering as well as aligning it with business network design. For this purpose this research paper discusses different techniques of service identification and design and presents two techniques and its instruments how a business driven discovery of services can enhance the financial networks design. The Swiss Banking sector serves to motivate and demonstrate the applicability of the suggested model due to the ongoing structural transformation driven by competence orientation, increased competition and business model adjustment.

**Keywords:** service-oriented architecture, service identification and clustering, business network redesign, service map

## 1 Introduction

### 1.1 Motivation

Following the tradition of object- and component-oriented architecture models, the service-oriented architecture (SOA) concept promises on the first hand as a technological concept the integration of heterogeneous application environments. However, SOA can also contribute to a more flexible allocation of business activities among partners in a value chain or network. This requires for adequate integration between the technological and the business world. In many contributions and discussions SOA is attributed a ‘silver bullet’ status to reach these goals. The key element and basic precondition for implementing SOA and providing the critical link to the business processes is the identification and clustering of business functions as services. For this purpose this paper will exemplify how serviced can be deduced and composed in a business-driven manner and verified by business as well as technical oriented design principles and criteria. Furthermore it will be outlined how the proposed techniques fit in

an engineering-oriented framework supporting business network redesign (BNR). The objective is supporting BNR by different instruments, guidelines and procedure models. This paper belongs to a multilateral, two-year research program that started in summer 2006 and investigates the management of service-oriented networks in the banking industry succeeding a completed two-year research program about bilateral sourcing. Our 18 research partners cover various institutional sizes and roles in the banking value chain (e.g. regional retail bank, international private bank, outsourcing provider, software provider). Beside specific bilateral projects, the partners contribute to the research in biannual steering committee meetings and quarterly workshops supplemented by case studies and interviews taking place throughout the research program substantiating the applicability of the envisioned approach. Within this research program this paper focuses on service identification and clustering as well as the instruments service map and service cluster and exemplifies their application in the payments process. This paper follows the argumentation of Steen et al. claiming that SOA “provides better handles for architectural alignment and business and IT alignment, in particular” [St05]. Moreover we argue that the concept of service-orientation can be used as well to foster BNR.

## **1.2 Research Focus**

By elaborating a methodology/architecture this research adopts a design science approach [He04] and presents results which have been elaborated in this research program. This comprised four workshops with all partners and 19 bilateral semi-structured interviews. The artefact, which is designed in this paper, is a technique for identifying and clustering business-driven services and a vertical consolidation with design instruments for business network redesign. A unified methodological approach for BNR on all layers has not been reached yet even though BNR is in debate for several years. Moreover as to be shown there is a lack of methodologies for service modelling aligned with sourcing models and combined with instruments for BNR. Business transformation, currently expressed by the integration of applications and the networking among companies (business networking) is apparent in the financial industry. Contrary to other industries such as the automotive industry most European banks developed proprietary applications over the last decades. This resulted in complex, heterogeneous and monolithic application landscapes with numerous proprietary interfaces and an increased total cost of ownership [HRW04]. As stated in several interviews with bank representatives during our research, many banks therefore aim at introducing standardized application architectures which may be maintained on a modular basis from a third party. The banking industry is facing a growing need to reduce vertical integration and the necessity to tap the potential of specialization effects in business networking. The industrialization of the finance industry as well as the emergence and redesign of networks such as the three networks grouped around the service provider Finnova, Avaloq and RTC (Real-Time-Center), initiated by Swiss cantonal banks, is currently in progress and requires adequate and business aligned application architectures to manage the growing complexity [Kn06]. The Swiss and German Banking sector and especially the payments process has therefore been chosen to motivate and demonstrate the applicability of the suggested model.

The structure of the paper reflects his goals. Subsection 2.1 and 2.2 discusses methodologies and concepts for business transformation and enterprise architecture, followed by subsection 2.3 describing drivers and challenges of service-oriented architectures. Subsection 3.1 carries out existing strategies for service modelling and based upon this foundation subsection 3.2 and 3.3 elaborate the integration of SOA and service modelling in BNR as well as conceive a hybrid technique for service identification and clustering. The functionality and the applicability of the proposed approach will be exemplified in section 4 at the cases of Equens and Postbank. The paper concludes with subsection 5 and a discussion of potential weaknesses and further research.

## **2 Services and Business Transformation**

### **2.1 Methodologies for Business Transformation**

Based upon drivers such as globalization, innovation and an increase in market competition, business transformation towards more decomposition, disintegration and networking has been recognized in many industries. Currently it is evolving in the banking industry [GH03] especially in Swiss and German institutes, which is one reason why the two payment processing companies Equens and Postbank has been chosen as case study in this paper. While business transformation is a key theme, it has already been pursued by business process redesign (BPR) and business network redesign (BNR). E.g. Venkatraman [Ve94] conceived the redesign of (external) business networks as logically next step after the redesign of cross-functional processes inside an organization.

Following Alt [Al06], models are important instruments for reducing complexity and distinguishing various elements on several interconnected layers as part of a BNR methodology. Existing enterprise modelling approaches, such as Multi-Perspective Enterprise Modelling (MEMO), Semantic Object Model (SOM) or Architecture for Integrated Information Systems (ARIS) follow this principle. Most of these methodologies have emerged with process-orientation and conceive processes as links between business strategies and the (technological) application architecture. Approaches such as Business Engineering (BE) [Oe95] that aim at semi-formalization of procedures, roles, activities and result documents have been termed engineering-like methodologies recognizing as well the business process as main lever of change and therefore key element in shaping future business solutions and the underlying IS [Oe01]. As procedures, activities and result documents are in the focus of this research and services are conducted in a business-driven manner the 'Business Engineering Model' (BE) (see [Oe95]) has been chosen as foundation, simultaneously providing consistency across the three layers: strategy, process and systems.

### **2.2 Enterprise Architecture**

Existing enterprise architecture approaches [Fo03] are focusing on processes, objectives and organizational structures and deduce business requirements for systems design

lacking in terms of cross-enterprise processes and networkability. Similarity can be recognized in approaches of organizational architecture [BSZ01, 267ff.] focusing on distribution of decision rights and incentive systems. ANSI/IEEE [Ie00] is defining enterprise architecture as organization of a system implying its components, relationships and governance structures. Enterprise architecture frameworks provide meta models, design methods, common vocabulary and reference models. As referred to in subsection 2.1 the BE has been chosen to provide structure to the approach in this paper.

### **2.3 Service-orientation and Business Transformation**

SOA is recognized as an important concept for business transformation and is discussed from two perspectives (technological, business). Nevertheless SOA has like many 'magic words' numerous different definitions. For example, SOA is conceived in a technical view as a "paradigm that supports modularized exposure of existing application functionality to other applications as services" ([Na04], 41). SOA in a broader view can be defined as the "policies, practices, frameworks that enable application functionality to be provided and consumed as sets of services published at a granularity relevant to the service consumer" [SW04, 3]. Service-orientation from a business view denotes the ability of reusing tasks and processes by solving them at one location [KÖ06, 236]. Therefore SOA has been proposed as dedicated layer between processes and systems for several business transformation frameworks.

Core element of any SOA are specified services, which may be identified in general by two approaches: technical-driven service modelling (bottom-up) and business-driven service modelling (top-down). For a combination of bottom-up and top-down the term hybrid has been suggested by [KKB07]. Procedure models for all approaches will be described and distinguished in section 3. Due to the fact that a general definition of services as part of a SOA is missing (see [FS05, 756]) and the aim of the paper is providing a business-driven service approach, services will be defined as: "independent usable and extensive specified functional components, which support the value performance of process activities".

A reduction in operating risks, time-to-market, integration costs and maintenance costs are only few benefits ascribed to SOA. Contrary higher complexity is suspected. In order to reduce complexity a classification framework for SOA and services should be provided. Based upon prior research (see [AGL05], [Sa05], [Ta05]) three service layers has been differentiated and comprises (1) process services which support activities of the core processes of a company and include some references to at least one activity of a business process such as foreign currency supply service and regulation service, (2) rule services which encapsulate business and validation rules used by process services such as product rule service and regulation rule service, and (3) entity services which encapsulate core entities and business objects, such as contract, partner or order. Infrastructure services providing services of a fine granularity to support transportation of information at data level are outside the scope of this paper.



### **3 Towards Architecture for Service-orientation**

Based upon this foundation the following chapter will differentiate related work by comparing the derived strategies of service modelling: top-down, bottom-up and hybrid. Subsection 3.2 will disclose the integration of the instruments service map and service cluster with instruments on the process and strategy layer followed by the exemplification of hybrid service identification and clustering techniques exceeding existing approaches.

#### **3.1 Comparison of Existing Research Approaches**

As described above, service modelling based upon a top-down approach is mainly used when understanding SOA as a concept of connecting business and technology. Based upon the analysis of business processes or business events [KKB07], service candidates are identified by applying widespread design principles ([Ba05], [Fr04] and [PG03]): loose coupling, modularity, business orientation and interface orientation. Existing business processes are decomposed to achieve service candidates. However a pure top-down approach neglects addressing the underlying and existing IS applications. Though services using a top-down approach provide a proper support for modelling new business roles such as global custodian or credit factory by orchestrating services, existing IS applications and platforms need to be taken into account in order to reduce setup costs and verify technical feasibility. Contrary, core element and basis for the bottom-up approach of service modelling are existing applications. A key step within provided procedure models is the analysis of currently existing applications and their IS functionality [Na04] as foundation for systems reengineering [ZLY05]. Researchers of bottom-up service modelling such as [Na04], [KSR04] or [ZLY05] are focusing on consolidating and rationalizing access to IS functionality by using services. [Na04] e.g. argues that technology-based and application-based composition of services can provide broader benefits than business-driven service composition, as technology's capabilities and back-end systems may be used more efficient and effective. The achievable benefit and key driver for bottom-up service modelling can therefore mainly be seen in the application integration of heterogeneous landscapes as well as in reduction of maintenance costs. Services and SOA are used to consolidate "multiple applications running on varied technologies and platforms" [Na04, 41]. However numerous application strategies besides SOA already exist and the necessary alignment of business processes and IS applications as basis for faster time-to-market and more flexible business models are not addressed in this approach. Nevertheless as top-down service modelling is focusing on existing business processes and bottom-up service modelling is based upon existing applications a third approach has emerged to capture functionality contained neither in processes nor in applications. Middle-out service modelling or goal modelling is described e.g. by [LA02], [Ar04] or [Sa05]. Business is modelled as goals and sub-goals, underlined by key performance indicators and metrics representing the quality of the so reached service candidates. Services are identified and modelled focusing on these goals. However the challenge remains to identify the cut of the business goals and to ensure the fit with the remaining enterprise architecture. To comprise the existing approaches the criteria strategy, origin and examination of the service cut are discussed as they occur in all methods. Moreover existing approaches can

be criticized lacking in terms of visualization, categorization and incorporation with instruments on process and strategy layer resulting in the next criteria shown in table 1. Simultaneously the criteria were iteratively discussed in the mentioned workshops and following the requirements of business transformation and the claim for network design within an engineering framework.

	KKB07	Na04	Ar04, LA02	Presented approach
Strategy	Top-Down	Bottom-up	Middle-out	Hybrid
Origin	business process	existing application	business intents	business process, network and sourcing model
examination of the service cut	design principles, stakeholder	application functionality	design principles	design principles, sourcing models, reference processes,
visualization instruments	-	-	goal service graphs	service map, service cluster
alignment with instruments on process and strategy layer	-	-	-	reference processes, role models, reference networks
service categorization	process-, basic services	-	-	process-, rule-, entity services
service composition / clustering	via process services	-	via enterprise components	via service clusters
service specification	operation, input, output, consumer	-	-	e.g. description, input, output, serviceuser, business object
application of process models	business process	-	-	reference and existing business processes
application of sourcing models	-	-	-	incorporation of developed sourcing models for the deduction step
reference to network and business model design	-	-	-	Incorporation of developed network and existing business models

Table 1: Comparison of existing service modelling strategies

Currently evolving initiatives such as the Industry Value Network of SAP try to combine the recognized lack of combining business-driven service identification and clustering with technical feasibility. However a methodology which above all is integrated in an engineering framework, linked with instruments and procedure models for business network redesign and taking sourcing models and business roles into account has not been reached yet.

### 3.2 Integration of Network Design and Service Modelling

As business processes are within the BE the main lever, the services within our approach are further based upon (reference) sourcing models (cf. figure 1). The so reached services are composed to service clusters which on one side provide more flexibility to disaggregated business processes providing the missing link between existing

application landscapes and business and on other side interrelate with business roles elaborated by business models and networks. Business roles, represented in a role model, can apply the service-orientation paradigm and interrelate directly with the service clusters. The business roles, representing certain business models, such as research provider, product designer, asset manager, valor data refiner or global custodian use or provide certain service clusters designed as independent from each other.

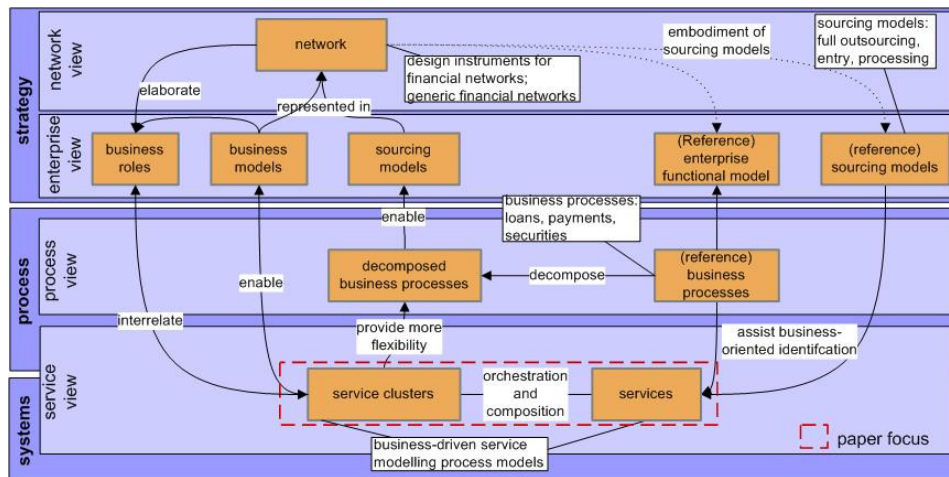


Fig.1: instruments for business network redesign

As the service cut is based upon these sourcing models, business processes and business roles the incorporation of legal requirements such as customer data access increases the reusability of the specified services by enhancing the ability to support diversified business strategies (scope and scale) to the same extent. The analysis of business networks is enriched by the embodiment of used services and service clusters. The instrument of the service map affiliates the approach of reducing complexity by structuring the services clusters in two dimensions: customer proximity as well as core vs. support activity, resulting in three overlapping domains: distribution competence, execution competence and support competence (cf. figure 3). The service map contains the service clusters and its encapsulated services.

Linking sourcing models, business roles, processes and services within an engineering-oriented framework provide benefits for both sides: the challenge of the service cut is addressed as the cut last not longer solely on business processes and critics of existing BNR approaches addressing network modelling solely on high level without interdependency towards IS are prevented as service clusters and maps provide a connection to IS-models.

### 3.2 Procedure Model for Service Identification and Clustering

To avoid missing service candidates as described in the middle-out approach while simultaneously using the benefits of correlating business and IT with the hybrid approach, an engineering-based methodology covering both aspects is needed.

Enterprise architectures addressing business transformation can provide a foundation for business-driven service identification.

The proposed model extends existing approaches ([KKB07], [Ar04], [LA02]) by combining service identification and clustering, integrating it in an engineering-oriented framework and implying besides business processes, strategic aspects. Pattern such as design pattern or architectural pattern are used to structure e.g. communication elements or software systems in object-orientation [SB03] and can therefore be adapted to enhance the structure in SOA. Service clusters ensue this pattern paradigm by structuring services and visualization instruments such as service maps. The proposed techniques for service identification and clustering, shown in figure 2, consist of four phases covering preparation and initialization, analysis, verification and detailing. The differentiation of the four phases has been made on basis of the gradation of existing procedure models (e.g. [Ar04], [KKB07]) and has been verified in workshops and interviews. The cross-reference models, which are provided for the finance industry case within the methodology, are indicated in figure 2.

During the *preparation phase* the required models are selected and the area for service identification and clustering is identified. Besides the enterprise model, which describes all existing processes of a company on a high granularity, and the network model, the (reference) business process (in this paper the payments process) is needed for the identification of the services and the service list containing the specified services is needed for the clustering of the specified services. All three were elaborated in the mentioned research program. The network model with its described roles and therefore implicit exhibited business models is necessary in order to assess the service cut. During the *analysis phase* the service identification follows a top-down approach based upon the business processes and the network model either representing an as-is or to-be state. The service candidates are deducted through the fine granular activities of the process incorporating existing design principles as stated above and defined service criteria: specified service context, part of one service layer according to a specified service classification pattern, reusability level as well as defined status before and after a request.

A workflow is created based upon the business processes exemplifying the activities underlined with additional determined and associated information concerning:

- the state changes of the information and business objects (e.g. create, access)
- legal requirements concerning availability and ownership of data
- interdependencies of certain tasks and business rules
- business roles and sourcing strategies using/providing the activity

The deduction of the service clusters is based upon the service map and list corresponding to the analysed area and follows therefore a bottom-up approach. Again design principles and criteria are incorporated in the analysis. The result of the analysis phase is service and cluster candidates containing domain specific knowledge as in this paper of the finance industry.

After a service or cluster candidate has been identified a functional description has to be made. The *verification phase* examines if the candidates fulfil the criteria and design principles, if the functionality isn't already be provided by another service or incorporated in another cluster and if the candidate fits the business needs of the process and the role. Beside the criteria the candidates have been verified especially in terms of

technical feasibility in workshops and semi-structured interviews with business and application architects of our research partners as proposed in the Business System Planning method, also used for business component design by Albani [A103].

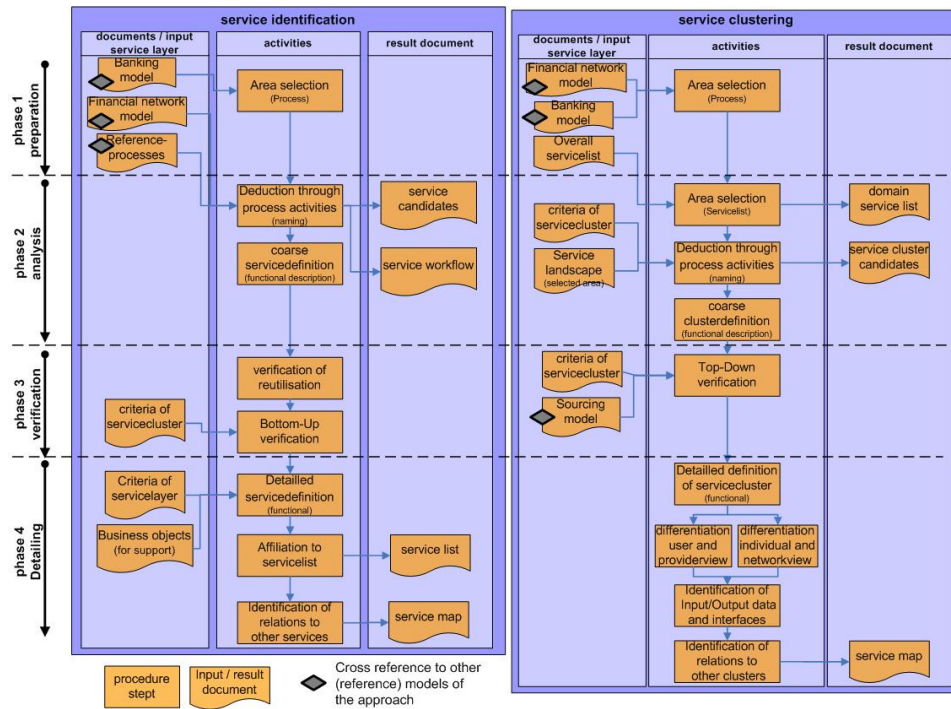


Fig.2: process models for business-oriented service identification and clustering

The assignment of the *detailing phase* is to specify the service or cluster in detail. The service is allocated according to a classification scheme including the following requirements: service functionality (results), service context (business object, classification), service behaviour (pre- and post condition, service interdependencies), service interface (input and output data), service quality (expected response time, automation, error recovery) and business impact (reusability, covered business tasks). The so-reached specification is comprehensible e.g. to Albani [A103] differentiating seven levels for the specification of business components and web services such as quality, behaviour and interface. Moreover during the service identification and clustering process guidelines and best practices have to be taken into account in order to provide an efficient service cut. Table 2 exemplifies some guidelines, which were developed and verified in several workshops with practitioners.

Guideline	Description
differentiation of rules	institute particular and legal rules are concentrated in rule services and allocated to process services
business-orientation	service identification is based upon as-is or to-be business processes detailed towards activities of a specific enterprise, business network and sourcing model or to-be reference business processes, reference business networks or reference

	sourcing models
usage of reference models	domain specific reference models have to be taken into account to avoid solely as-is modelling and sequential analysis
business-object intersection	data services always allude to one business object
incorporation of legal requirements	legal domain related rules have to be taken into account to support different business models, roles and sourcing models
data ownership	when orchestrating and re-using services, data ownership and availability has to be considered

Table 2: extracted guidelines for service design

## 4 Application of Methodology in the Payments Process

The following section will apply the techniques to the domain of the finance industry especially the payments process. As the payment process covers numerous activities we confine the analysis scope in subsection 4.1 upon the process step payments entry focusing the business object payments. Subsection 4.2 will exemplify service maps for two payment processing providers. The transformation of the European finance industry can be outlined by main drivers currently stressing the institutes, shown in table 3.

Driver	Impact
market changes	increased competition based upon globalization and changes in market structures (e.g. cantonal banks overcome provincial borders) as well as market concentrations [GH03]
Regulations	increased regulation efforts based upon emerging international guidelines such as SEPA (single European payment area)
customer structure	increased customer expectations based upon internet based banking solutions such as online brokerage
product complexity	increased product diversity result in higher costs for product listing [Kn06]
Technology	former development of proprietary applications resulted in intricate serviceable, mainframe-based and monolithic application landscapes [HRW04]
Competitiveness	decreasing margins based upon additional cost pools result in downward cost income ratios.

Table 3: main drivers for business transformation in the finance industry

Summarizing, the banking sector is facing currently two main challenges: application integration and value chain reconfiguration [Ba05]. Furthermore the required business networking based upon competence orientation and diversification is inadequately supported by existing core banking platforms as they facilitate networkability only to a low extent [AS07]. According to the apparent business transformation in German and Swiss banks, the finance industry has been chosen to exemplify the applicability of the proposed techniques.

### 4.1 Service Specification

Following the proposed techniques and guidelines using banking-specific models (payment reference role model and business network, payment reference business process and payment reference sourcing models), 48 services and 17 clusters can be identified. Table 4 exemplifies at the payments data service how these 48 services were specified resulting in one service list. Afterwards the services and clusters were exhibited in a service map. According to the to-be business process which was used the

interdependencies between the services were examined and also incorporated in the service map. Since, the service map is used to enhance the design of different business models, represented as a business role, such as specialist execution, foreign currency trader or specialist regulations. The service clusters are used at first sight, to describe the underlying functionality (business scope). Simultaneously existing business models, such as of the Swiss private bank Vontobel acting as portfolio manager for the Raiffeisen Classic Portfolio of the Swiss association of Raiffeisenbanken (SVRB) can be easier analysed as service clusters provide the often missed link between strategy and IS.

section	Subsection	Description
service functionality	Results	Service provides necessary information of the customer throughout the transaction, where customer master data can't be accessed directly due to legal requirements. Service encapsulates all data relevant for the execution of a payment and enables a one-view to one transaction including the status.
service context	business object	single payment
	Classification	data service layer
service behaviour	pre condition	ID available and existing
	post condition	datasets are readout and returned
	service interdependencies	used by process services in the payment execution process
service interface	Input	payment-ID
	output data	account number, customer master data, instrument, execution date, currency, customer discount ...
service quality	expected response time	less than 2 seconds
	Automation	fully automated; STP-rate 100%
	error recovery	level A; within 1hour
business impact	Reusability	reutilization where transactions are processed
	covered business tasks	throughout the business process, where transaction data is accessed

Table 4: extracted specification of payment data service

#### 4.2 Application of Service Design in Two Cases

The enhancements for the design of networks on the basis of the proposed service modelling approach are illustrated by two cases: Deutsche Postbank AG (DPB) and Equens N.V. (Equens), which are major players for payment execution in Germany. Both provide sourcing models for the execution of domestic payments as well as support services for archiving, investigations and control. Furthermore the offer of DPB covers the execution of foreign country payments, regulation examination and foreign currency trade. In terms of sourcing levels, DPB offers almost full outsourcing and Equens offers a sourcing model based upon payment execution. Therefore both business models encompass the same core (payment execution) but differ in terms of scope. DPB and Equens are operating within a widespread correspondence network and cooperate with clearing institutes, national banks, distribution banks and others. Moreover payments execution is a standardized business with low margins, which results in high potentials for outsourcing. Figure 3 exemplifies the reduced service maps of both providers focusing only on the service clusters. Though the differences are minor at first sight, the implications by looking at the offered services are more fundamental. Concentrating on the payments entry cluster, which consists of 15 services, Postbank offers all services in contrary to Equens, which doesn't offer digitalization of non-electronic payment

instructions, supported by the recognition, digitalization and recognition rule service. The identified services can support both providers to the same extent and the proposed reference service clusters can avail the analysis of business models by detailing them via service clusters and services. Additionally the communication between IT and business department as well as between enterprises is enriched by specified and standardized service elements on different abstraction layers providing a clear link between strategy, process and systems layer.

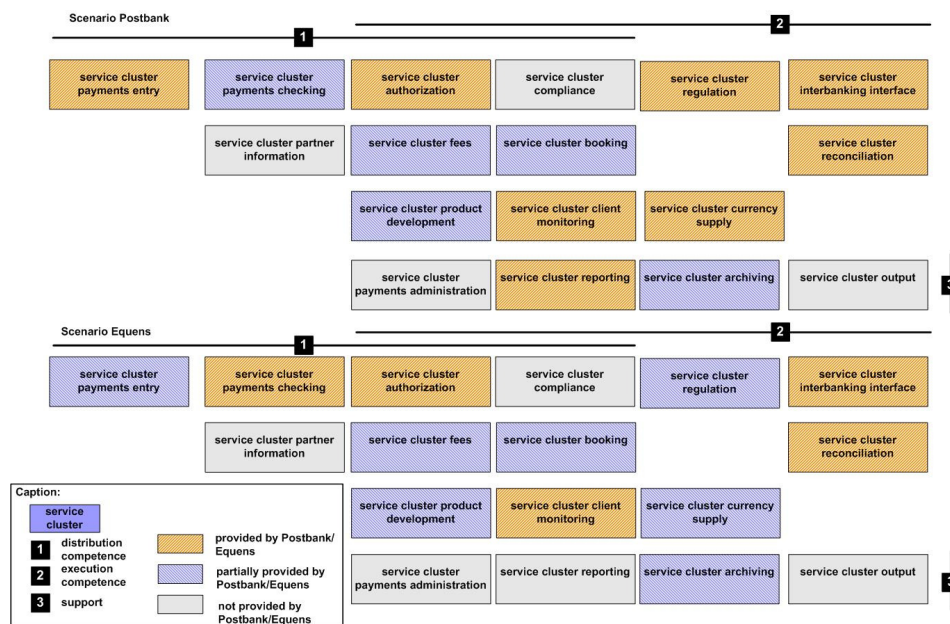


Fig. 3: service cluster maps Postbank and Equens

## 5 Summary and Outlook

Swiss banks are currently facing a fundamental transformation towards more networked structures (cf. section 1 and 4). In order to reach high efficiency and flexibility the redesign of networks should be supported by an integrated methodology implying procedure model, guidelines and instruments as well as standardization efforts. Service orientation and SOA seems to be an adequate ‘silver bullet’ to support the business transformation (cf. section 2.3). Key element in implementing SOA is the identification and clustering of business functions as services (cf. section 2.3). However by linking it with strategic aspects existing approaches show shortcomings (cf. section 3.1). The paper has therefore presented an approach for business-oriented service modelling (cf. section 3.3) as a combination of business-process driven and business-object driven service identification and technical verification by design principles and service criteria supplemented by a process model for service clustering. Coinstantaneous the service-oriented architecture concept has been integrated in an engineering-oriented framework (cf. section 3.2). The instrument of the service map has been deduced as a result of the



techniques (cf. section 3.2). The case studies Postbank and Equens have been used to apply the service map (cf. section 4.2). Standardized and specified business oriented services bypass differences in terms of business model scope and business process sequence.

Further research should address a detailing of the guidelines and the formulation of the procedure model for BNR, including the presented techniques. The holistic methodology will be based upon a meta model, which is in progress and will be presented in one of the next papers. Moreover we will dare to apply the service-orientation paradigm towards the strategy layer by converging business roles and service clusters aiming at a consistently enterprise architecture. By the time we will apply the instruments to further industries in order to provide a generalized methodology. Furthermore the techniques should be enhanced by the aspects of service versioning and iteratively design and redesign of as-is and to-be services. Coinstantaneous we will focus on how different service maps of e.g. providers and banks can be examined and matched.

## References

- [AI03] Albani, A. et al.: Identification and Modelling of Web Services for Inter-enterprise Collaboration Exemplified for the Domain of Strategic Supply Chain Development, In: Proceedings of the OTM Confederated International Conferences CoopIS, DOA, and ODBASE 2003, Catania, 2003; pp. 74-92.
- [AI06] Alt, R.: Business Network Redesign – Overview of Methodologies and Example of Process Portals. In Business Process Transformation (Markus, L.M., Grover, V., Series: Advances in Management Information Systems, M.E. Sharpe) (2006).
- [AGL05] Alt, R.; Gizanis, D.; Legner, C.: Collaborative Order Management: Towards Standard Solutions for Interorganisational Order Management. In: International Journal Technology Management, 31 (1/2), 2005; pp. 78-97.
- [AS07] Alt, R.; Smits, M.: Networkability of Organizations and Business Networks, In: ECIS'07: Proceedings of the 15<sup>th</sup> European Conference on Information Systems, St. Gallen, 2007 pp. 119-130.
- [Ar04] Arsanjani, A.: Service-oriented Modelling and Architecture, In: IBM DeveloperWorks, 2004
- [Ba05] Baskerville, R. et al.: Extensible Architectures: The Strategic Value of Service-Oriented Architecture in Banking. In: ECIS'05: Proceedings of the 13th European Conference on Information Systems, Regensburg, 2005, pp. 761-772.
- [BSZ01] Brickley, J.A., Smith, C.W., Zimmerman, J.L.: Managerial Economics and Organizational Architecture, 2. edition., McGraw-Hill, Boston 2001.
- [Fr04] Fritz, F.-J.: An Introduction to the Principles of Enterprise Services Architecture (ESA), In: SAP Insider, 2, 2004; URL: [http://www.sapinsideronline.com/spijsp/article.jsp?article\\_id=37906&volume\\_id=5147](http://www.sapinsideronline.com/spijsp/article.jsp?article_id=37906&volume_id=5147), (26.04.2007).
- [FS05] Ferguson, D.; Stockton, M.: Service-oriented architecture: Programming model and product architecture, In: IBM Systems Journal, 44 (4), 2005; pp. 753-780.
- [Fo03] Foegen, M., Architektur und Architekturmanagement - Modellierung von rchitekturen und Architekturmanagement in der Softwareorganisation, in: HMD - Praxis der Wirtschaftsinformatik, 40 (2003) 232, S. 57-65
- [GH03] Geiger, H., Hürzeler, H.: The Transformation of the Swiss Private Banking Sector, In: Journal of Financial Transformation, 9, 2003; pp. 93-103
- [He04] Hevner, A.R. et. al.: Design Science in Information Systems Research, In: MIS Quarterly 28 (1), 2004; pp. 75-105.

- [HRW04] Homann, U.; Rill, M.; Wimmer, A.: Flexible Value Structures in Banking, In: Communications of the ACM, 47 (5), 2004; pp. 34-36.
- [Ie00] IEEE (eds.): IEEE Recommended Practice for Architectural Description of Software Intensive Systems, IEEE Std 1471-2000, 2000. Available: <http://standards.ieee.org/reading/ieee/std/se/1471-2000.pdf>
- [KKB07] Klose, K., Knackstedt, R., Beverungen, D.: Identification of Services – A Stakeholder-based Approach to SOA Development and its Application in the Area of Production Planning, In: ECIS'07: Proceedings of the 15<sup>th</sup> European Conference on Information Systems, St. Gallen, 2007; pp. 1802-1814.
- [Kn06] Knowledge@Wharton (Eds.): Special Report: Unraveling Complexity in Products and Services, Philadelphia, 2006; pp. 1-12.
- [KÖ06] Kagermann, H.; Österle, H.: Geschäftsmodelle 2010. Wie CEOs Unternehmen transformieren, 2<sup>nd</sup> edition, Frankfurt, 2006.
- [KSR04] Kaabi, R. S.; Souveyet, C.; Rolland, C.: Eliciting Service Composition in a Goal Driven Manner, In: ICSOC '04: Proceedings of the 2<sup>nd</sup> international conference on Service oriented computing, New York, 2004; pp. 308-315.
- [LA02] Levi, K., Arsanjani, A.: A Goal-driven Approach to Enterprise Component Identification and Specification, In: Communications of the ACM, 45 (10), 2002; pp. 45-52.
- [Na04] Nadham, E.G.: Seven Steps To a Service-Oriented Evolution, In: Business Integration Journal, 2004; pp. 41-44.
- [Oe95] Oesterle, H.: Business in the Information Age. Berlin etc.: Springer, (1995) pp. 13-18.
- [Oe01] Oesterle, H.: Enterprise in the Information Age. In Oesterle H.; E. Fleisch, R. Alt (Eds.), Business Networking: Shaping Collaboration Between Companies. Berlin etc.: Springer, (2001) pp. 17 - 53.
- [PG03] Papazoglou, M.P.; Georgakopoulos, D.: Service-oriented Computing, In: Communications of the ACM, 46 (10), 2003; pp. 25-28.
- [Sa05] SAP (Eds.): Enterprise Services Design Guide, SAP AG, 2005; URL: [http://www.sap.com/platform/netweaver/pdf/BWP\\_ES\\_Design\\_Guide.pdf](http://www.sap.com/platform/netweaver/pdf/BWP_ES_Design_Guide.pdf) (12.02.2007).
- [SB03] Schmidt, D.C., Buschmann, F.: Patterns, Frameworks, and Middleware: Their Synergistic Relationships, In: ICES'03: Proceedings of the 25<sup>th</sup> International Conference on Software Engineering, Portland, 2003; pp. 694-704.
- [St05] Steen, M.W.A. et al.: Service-Oriented Enterprise Architecture, In: (Stojanovic, Z.; Dahanayake, A.), Service Oriented Systems Engineering, Hershey, 2005; pp. 132-154.
- [SW04] Sprott, D.; Wilkes, L.: Understanding Service-Oriented Architecture, In: The Architecture Journal, 2004.
- [Ta05] Taylor, J.: Achieving Decision Consistency across the SOA-based Enterprise Using Business Rules Management Systems. In: WISE '05: Proceedings of the Web Information Systems Engineering Conference, New York, 2005; pp. 750-761.
- [Ve94] Venkatraman, N.: IT-Enabled Business Transformation: From Automation to Business Scope Redefinition, MIT Sloan Management Review, Vol. 35, No. 2, pp. 73 – 87 (1994).
- [ZLY05] Zhang, Z.; Liu, R.; Yang, H.: Service Identification and Packaging in Service Oriented Reengineering, In: SEKE '05: Proceedings of the 17<sup>th</sup> International Conference on Software Engineering and Knowledge Engineering, 2005.