

Gesellschaft für Informatik e.V. (GI)

publishes this series in order to make available to a broad public recent findings in informatics (i.e. computer science and information systems), to document conferences that are organized in co-operation with GI and to publish the annual GI Award dissertation.

Broken down into the fields of

- Seminar
- Proceedings
- Dissertations
- Thematics

current topics are dealt with from the fields of research and development, teaching and further training in theory and practice. The Editorial Committee uses an intensive review process in order to ensure the high level of the contributions.

The volumes are published in German or English

Information: <http://www.gi-ev.de/service/publikationen/lni/>

ISSN 1617-5468

ISBN 978-3-88579-246-8

This volume contains the Proceedings of the 3rd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 2009) which 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). It features a selection of 11 high quality contributions from academia and practice on Information Systems, Business Informatics and Computer Science. Leading authors from academia and industry present state-of-the-art developments, experiences, and best practices in this volume.



J. Mendling, S. Rinderle-Ma, W. Esswein (Eds.): EMISA 2009

GI-Edition

Lecture Notes in Informatics

**Jan Mendling, Stefanie Rinderle-Ma,
Werner Esswein (Eds.)**

Enterprise Modelling and Information Systems Architectures

**Proceedings of the 3rd Int'l Workshop EMISA
2009, Ulm, Germany**

September 10–11

Proceedings





Jan Mendling, Stefanie Rinderle-Ma, Werner Esswein (Eds.)

Enterprise Modelling and Information Systems Architectures

Proceedings of the 3rd International Workshop on Enterprise Modelling and Information Systems Architectures

**Ulm, Germany
September 10-11, 2009**

Gesellschaft für Informatik e.V. (GI)

Lecture Notes in Informatics (LNI) - Proceedings

Series of the Gesellschaft für Informatik (GI)

Volume P-152

ISBN 978-3-88579-246-8

ISSN 1617-5468

Volume Editors

Prof. Dr. Jan Mendling

Humboldt-Universität zu Berlin

10178 Berlin, Germany

Email: jan.mendling@wiwi.hu-berlin.de

PD Dr. Stefanie Rinderle-Ma

Universität Ulm

89069 Ulm, Germany

Email: stefanie.rinderle@uni-ulm.de

Prof. Dr. Werner Esswein

Technische Universität Dresden

01062 Dresden, Germany

Email: werner.esswein@tu-dresden.de

Series Editorial Board

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

Hinrich Bonin, Leuphana-Universität Lüneburg, Germany

Dieter Fellner, Technische Universität Darmstadt, Germany

Ulrich Flegel, SAP Research, Germany

Ulrich Frank, Universität Duisburg-Essen, Germany

Johann-Christoph Freytag, Humboldt-Universität Berlin, Germany

Thomas Roth-Berghofer, DFKI

Michael Goedicke, Universität Duisburg-Essen

Ralf Hofestädt, Universität Bielefeld

Michael Koch, Universität der Bundeswehr, München, Germany

Axel Lehmann, Universität der Bundeswehr München, Germany

Ernst W. Mayr, Technische Universität München, Germany

Sigrid Schubert, Universität Siegen, Germany

Martin Warnke, Leuphana-Universität Lüneburg, Germany

Dissertations

Dorothea Wagner, Universität Karlsruhe, Germany

Seminars

Reinhard Wilhelm, Universität des Saarlandes, Germany

Thematics

Andreas Oberweis, Universität Karlsruhe (TH)

© Gesellschaft für Informatik, Bonn 2009

printed by Köllen Druck+Verlag GmbH, Bonn

Preface

The strategic importance of modelling is recognized by an increasing number of companies and public agencies. Enterprise modelling delivers the ‘blueprints’ for co-designing organizations and their information systems, such that they complement each other in an optimal way. Achieving this interplay requires a multi-perspective approach that takes into account technical, organizational, and economic aspects. It also recommends the cooperation of researchers from different fields such as Information Systems, Business Informatics, and Computer Science.

The 3rd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA’09) addresses all aspects relevant for enterprise modeling 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 high-quality contributions from academia and practice on enterprise modelling, enterprise architectures, business process management and measurements, analysis, and explorative studies. We received 22 submissions that were thoroughly reviewed by at least three selected experts of the programme committee. Eleven contributions were selected for presentation at the workshop and for publication in these proceedings.

We would like to thank the members of the programme committee and the reviewers for their efforts in selecting the papers and helping us to compile a high-quality programme. We would also like to thank the local organization, in particular Linh Thao Ly as well as the organization team of BPM 2009 for their assistance with organizing EMISA 2009. We also thank Manfred Reichert and Bertram Ludäscher as keynote speakers. We hope you will find the papers in this proceedings interesting and stimulating.

August 2009

Jan Mendling, Stefanie Rinderle-Ma, and Werner Esswein.

Programme Co-Chairs

Jan Mendling, Humboldt-Universität zu Berlin, Germany

Stefanie Rinderle-Ma, Universität Ulm, Germany

Werner Esswein, Technische Universität Dresden, Germany

Programme Committee

Witold Abramowicz, Poznan University of Economics

Antonia Albani, University of Augsburg/TU Delft

Colin Atkinson, University of Mannheim

Lars Bækgaard, Aarhus School of Business

Jörg Becker, University of Münster

Martin Bertram, Commerzbank Frankfurt

Peter Bøgh Andersen, Aarhus University

Jörg Desel, KU Eichstätt

Fernand Feltz, CREDI Luxembourg

Ulrich Frank, University of Duisburg-Essen

Andreas Gadatsch, FH Siegburg-Bonn

Claude Godart, Université Henri Poincaré Nancy

Wilhelm Hasselbring, University of Kiel

Brian Henderson-Sellers, University of Technology Sydney

Heinrich Jasper, Freiberg University of Technology

Reinhard Jung, Universität Duisburg-Essen

Dimitris Karagiannis, University of Vienna

Fredrik Karlsson, Örebro University

Roland Kaschek, Kimep Almaty

Stefan Kim, Universität Hohenheim

Ralf Klischewski, German University Kairo

Gerhard Knolmayer, Universität Bern

John Krogstie, University of Trondheim

Dominik Kuropka, HPI Potsdam

Susanne Leist, University of Regensburg

Stephen W. Liddle, Brigham Young University

Mikael Lind, University College of Borås

Peter Loos, Universität des Saarlandes

Bernd Müller, FH Wolfenbüttel

Klaus-Walter Müller, MSG Systems AG, München

Markus Nüttgens, University of Hamburg

Andreas Oberweis, University of Karlsruhe

Erich Ortner, TU Darmstadt

Sven Overhage, University of Augsburg

Hansjürgen Paul, Institut Arbeit und Technik

Erik Proper, Radboud University, Nijmegen

Michael Rebstock, University of Applied Sciences Darmstadt

Manfred Reichert, Universität Ulm

Peter Rittgen, Vlerick Leuven Gent Management School

Michael Rosemann, Queensland University of Technology

Matti Rossi, Helsinki Business School
Gunter Saake, University of Magdeburg
Eike Schallehn, University of Magdeburg
Guttorm Sindre, University of Trondheim
Elmar J. Sinz, University of Bamberg
Stefan Strecker, University of Duisburg-Essen
Klaus Turowski, University of Augsburg
Willem-Jan van den Heuvel, Tilburg University
Gottfried Vossen, University of Münster
Barbara Weber, Universität Innsbruck
Hans Weigand, Tilburg University
Mathias Weske, HPI Potsdam
Roel Wieringa, University of Twente
Robert Winter, University of St. Gallen

Additional Reviewers

Stephan Aier
Tayyeb Amin
Dominik Birkmeier
Elżbieta Bukowska
Marcus Elzenheimer

Hans-Georg Fill
Jacques Klein
Azeem Lodhi
Stefan Strecker
Stephan Vornholt

Local Organisation

Linh Thao Ly, Ulm University, Germany
Stefanie Rinderle-Ma, Ulm University, Germany



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) is aimed at providing a forum for exchanging ideas and solutions on modelling research within Information Systems - both for researchers at universities and 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.

Table of Contents

Keynote

| | |
|---|---|
| <i>A Thing Called “Fluid Process” – Beyond Rigidity in Business Process Support</i> Manfred Reichert | 9 |
|---|---|

Enterprise Modelling

| | |
|--|----|
| <i>Pattern Matching in Conceptual Modes– A Formal Cross-Modelling Language Approach</i> Patrick Delfmann, Sebastian Herwig, Lukasz Lis, and Armin Stein | 13 |
| <i>Integration of Object Oriented Domain Modeling and Meta-Modeling</i> Antoine Schlechter, Simon Guy, and Fernand Feltz | 27 |
| <i>Integrating System Dynamics with Object-Role Modeling and Petri Nets</i> Fiona Tulinayo, Stijn Hoppenbrouwers, Patrick van Bommel, and Erik Proper | 41 |

Measurements, Analysis and Explorative Studies

| | |
|---|----|
| <i>Application Landscape Metrics: Overview, Classification, and Practical Usage</i> Jan Stefan Addicks and Philipp Gringel | 55 |
| <i>How Do System and Enterprise Architectures Influence Knowledge Management in Software Projects? – An Explorative Study of Six Software Projects</i> Carsten Lucke, Markus May, Nane Kratzke, and Ulrike Lechner | 69 |
| <i>From Process to Simulation - A Transformation Model Approach</i> Oliver Kloos, Volker Nissen, and Mathias Petsch | 83 |

Business Process Management

| | |
|--|----|
| <i>Controlled Flexibility and Lifecycle Management of Business Processes through Extensibility</i> Soeren Balko, Arthur H.M. ter Hofstede, Alistair Barros, Marcello La Rosa, and Michael Adams | 97 |
|--|----|

Process Flexibility: A Design View and Specification Schema 111
Udo Kannengiesser

Access Control for Monitoring System-Spanning Business Processes in Proviado 125
Sarita Bassil, Manfred Reichert, and Ralph Bobrik

Enterprise Architectures

A Survival Analysis of Application Life Spans based on Enterprise Architecture Models 141
Stephan Aier, Sabine Buckl, Ulrik Franke, Bettina Gleichauf, Pontus Johnson, Per Närman, Christian Schweda, and Johan Ullberg

Evaluating Enterprise Architecture Management Initiatives – How to Measure and Control the Degree of Standardization of an IT Landscape 155
Matthias Brückmann, Klaus-Manfred Schöne, Stefan Junginger, and Diana Boudinova

A Thing Called "Fluid Process" – Beyond Rigidity in Business Process Support

Manfred Reichert

Institute of Databases and Information Systems, University of Ulm, Germany
manfred.reichert@uni-ulm.de

Abstract: This keynote reports on a new class of processes - so called *fluid processes* - whose "engineering" and "use" is indistinguishable. Fluid processes are continually being adapted and reformed to fit the actual needs and constraints of the situation in hand and to fulfill the overall goals of the involved actors in the best possible way. We present a detailed review of challenges and techniques that exist for the support of fluid processes. We give insights into their nature, discuss fundamental challenges to be tackled, summarize basic technologies enabling fluid processes at the information system level, and describe advanced applications of fluid processes.

1 Motivation

The economic success of an enterprise increasingly depends on its ability to flexibly support its business processes by information systems and to react to changes in its environment in a quick and flexible way [RRMD09, WRRMW09]. However, today's process engineering technologies and tools are ill equipped to meet this challenge because of their inherent brittleness and inflexibility [WRRM08, RD98]. The current generation of BPM tools [Wes07] implicitly embrace the "engineer use" dichotomy inherited from traditional approaches to software and database engineering. This, in turn, is based on the classic engineering principle that systems are first "engineered" and then, once deemed fit for purpose, are "used" (or "operated"). Maintenance and evolution activities are not regarded as part of operation but rather as interruptions to the "in use" state which temporarily return the system to the "being engineered" state. However, in scenarios in which requirements come and go on a much more dynamic and ad hoc basis (e.g. healthcare [LR07] or automotive engineering [MHHR06]), this "engineer use" strategy is unworkable.

The only feasible way to cope with dynamism is to dissolve the fundamental distinction between "engineering" and "use" and seamlessly merge the whole service and process lifecycle into a single encompassing framework [WRRMW09, WSR09]. This will lead to a new class of processes - so called *fluid processes* - whose "engineering" and "use" is indistinguishable. Such processes are continually being adapted and reformed to fit the actual needs and constraints of the situation in hand and to fulfill the overall goals of the involved actors in the best possible way. Since enterprise workflows can be viewed as special forms of processes, the notion of fluid workflows is also important.

2 Characterization of Fluid Processes

Fluid processes and services should be not confused with “(self-)adaptive systems” as recognized by the adaptive systems research community. Fluid processes and services are adaptive in the sense that they are continually evolving and reshaping to fit the situation in hand, but unlike classical adaptive systems (as understood in adaptive system research) they are not expected to do this themselves. On the contrary the whole point of fluid process is that the adaptation is performed with the help of the human user/engineer where needed. In other words, in fluid processes, human engineers and users are “part of the loop”, and processes use and adaptation are seen as two sides of the same coin. In this sense, fluid process have more in common with agile software development methods, which are focused on encouraging human developers to evolve software in as rapid and effective a way as possible, than adaptive systems which are responsible for all dynamic adaptation themselves.

Adaptive systems research is still essentially based on the engineer – user distinction, the only difference to traditional approaches is that the systems are given much more intelligence than usual in order to exhibit a wider range of behaviors. This notwithstanding, the techniques and ideas of adaptive systems research are very relevant to fluid process and services, because all forms and techniques of adaptation need to be exploited. In fact, the notion of fluid process and services subsumes standard adaptive systems, and is based on a more general approach where humans are tightly bound into the loop as well as artificial intelligence.

Fluid processes also fundamentally change the way in which human stakeholders interact and collaborate because they dissolve the distinction between process engineers and process users. To date, business process support technologies have focused on enhancing and automating the way in which process users collaborate and interact, but have not significantly changed the way in which the processes themselves are engineered (i.e. defined and maintained). The assumption has been that this is done by IT specialists in a distinct engineering phase with little or no connection to the execution of the processes or the normal operation of the enterprise. However, with fluid processes this distinction will blur (if not entirely disappear) and process engineers will also be process users and vice versa. Stated differently, process engineering will also be regarded as a normal, fluid business process involving the collaboration of multiple stakeholders. In short, process engineering will become reflective.

3 Outline of the Keynote

This keynote presents a detailed review of challenges and techniques that exist for the support of fluid processes [RRMD09]. We give insights into the nature of fluid processes, discuss fundamental technological challenges to be tackled, give an overview on basic technologies enabling fluid processes at the information system level, and describe advanced applications of fluid processes. In this context we revisit some of our research projects

on fluid processes like ADEPT [RD98, RDB03, RRD03], CEOSIS [RMR07, RMR09], Corepro [MRH08], and Philharmonic Flows [KR09].

References

- [KR09] V. Künzle and M. Reichert. Integrating Users in Object-aware Process Management Systems: Issues and Challenges. In *Proc. Business Process Management Workshops 2009*. Springer, 2009.
- [LR07] Richard Lenz and Manfred Reichert. IT support for healthcare processes - premises, challenges, perspectives. *Data Knowledge Engineering*, 61(1):39–58, 2007.
- [MHHR06] Dominic Müller, Joachim Herbst, Markus Hammori, and Manfred Reichert. IT Support for Release Management Processes in the Automotive Industry. In *Business Process Management*, LNCS 4102, pages 368–377, 2006.
- [MRH08] Dominic Müller, Manfred Reichert, and Joachim Herbst. A New Paradigm for the Enactment and Dynamic Adaptation of Data-Driven Process Structures. In *Proc. CAiSE'08*, LNCS 5074, pages 48–63, 2008.
- [RD98] M. Reichert and P. Dadam. ADEPT_{flex} - Supporting Dynamic Changes of Workflows Without Losing Control. *JGIS*, 10(2):93–129, 1998.
- [RDB03] M. Reichert, P. Dadam, and T. Bauer. Dealing with Forward and Backward Jumps in Workflow Management Systems. *Software and Systems Modeling*, 2(1):37–58, 2003.
- [RMR07] S. Rinderle-Ma and M. Reichert. A Formal Framework for Adaptive Access Control Models. In *Journal of Data Semantics, IX*, LNCS 4601, pages 82–112, 2007.
- [RMR09] S. Rinderle-Ma and M. Reichert. Comprehensive Life Cycle Support for Access Rules in Information Systems: The CEOSIS Project. *Enterprise Information Systems*, 3(3):219–251, 2009.
- [RRD03] S. Rinderle, M. Reichert, and P. Dadam. Evaluation Of Correctness Criteria For Dynamic Workflow Changes. In W.M.P. v.d. Aalst, A.H.M. ter Hofstede, and M. Weske, editors, *Proc. Int'l Conf. on Business Process Management (BPM'03)*, LNCS 2678, pages 41–57, Eindhoven, The Netherlands, June 2003.
- [RRMD09] Manfred Reichert, Stefanie Rinderle-Ma, and Peter Dadam. Flexibility in Process-Aware Information Systems. *T. Petri Nets and Other Models of Concurrency*, 2:115–135, 2009.
- [Wes07] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.
- [WRRM08] Barbara Weber, Manfred Reichert, and Stefanie Rinderle-Ma. Change patterns and change support features - Enhancing flexibility in process-aware information systems. *Data Knowledge Engineering*, 66(3):438–466, 2008.
- [WRRMW09] Barbara Weber, Manfred Reichert, Stefanie Rinderle-Ma, and Werner Wild. Providing Integrated Life Cycle Support in Process-Aware Information Systems. *Int. Journal of Cooperative Information Systems*, 18(1):115–165, 2009.
- [WSR09] Barbara Weber, Shazia Wasim Sadiq, and Manfred Reichert. Beyond rigidity - dynamic process lifecycle support. *Computer Science - R&D*, 23(2):47–65, 2009.

Pattern Matching in Conceptual Models – A Formal Multi-Modelling Language Approach

Patrick Delfmann, Sebastian Herwig, Łukasz Lis, Armin Stein

University of Münster
European Research Center for Information Systems (ERCIS)
Leonardo-Campus 3
48149, Münster
Germany
{delfmann | herwig | lis | stein}@ercis.uni-muenster.de

Abstract: Recognizing patterns in conceptual models is useful for a number of purposes, for example revealing syntactical errors, model comparison, and identification of business process improvement potentials. In this contribution, we introduce a formal approach for the specification and matching of structural patterns in conceptual models. Unlike existing approaches, we do not focus on a certain application problem or a specific modelling language. Instead, our approach is generic making it applicable for any pattern matching purpose and any conceptual modelling language. In order to build sets representing structural model patterns, we define formal operations based on set theory, which can be applied to arbitrary models represented by sets. Besides a conceptual and formal specification of our approach, we present a prototypical modelling tool that shows its applicability through a particular application scenario.

1 Introduction

The structural analysis of conceptual models has multiple applications. To support modellers in their analyses, applying structural patterns to conceptual models is an established approach. Single conceptual models, for example, are analysed by use of typical error patterns in order to check for syntactical failures [Me07]. In the domain of Business Process Management (BPM), process models analysis helps identifying process improvement potentials [VTM08]. For example, applying structural model patterns to process models can help revealing changes of data medium during process execution (e.g., printing and retyping a document), redundant execution of process activities or application potentials of software systems. Whenever modelling is conducted in a distributed way, model integration is necessary to obtain a coherent view on the modelling domain. To find corresponding fragments and to evaluate integration opportunities, multiple models – generally of the same modelling language – can be compared with each other applying structural model pattern matching [GMS05]. Different model structures that typically represent equal real-world issues are identified and specified as structurally different, but semantically equal patterns. Counterparts of these patterns are

searched via pattern matching in the models to be compared. If pattern counterparts are found in different models, these are marked as candidates for equivalent model sections. A subsequent comparison of their elements shows if their contents are equal as well. This way, structural pattern matching provides decision support in model comparison and integration. Model patterns have already been subject of research in the fields of database schema integration and workflow management, to give some examples. However, a literature review reveals that existing pattern matching approaches are limited to a specific domain or restricted to a single modelling language (cf. Section 2). We argue that the modelling community would benefit from a more generic approach, which is not restricted to particular modelling languages or application scenarios.

In this paper, we present a set theory-based model pattern matching approach, which is generic and thus not restricted regarding its application domain or modelling language. We base this approach on set theory as any model can be regarded as a set of objects and relationships – regardless of the modelling language or application domain. Set operations are used to construct any structural model pattern for any purpose. Therefore, we propose a collection of functions acting on sets of model elements and define set operators to combine the resulting sets of the functions (cf. Section 3). This way, we are able to specify structural model patterns for a given modelling language in form of expressions built of the proposed functions and operators. These pattern descriptions can be matched against conceptual models of this language resulting in sets of model elements, which represent particular pattern occurrences. As a specification basis, we use a generic meta-meta model being able to instantiate any modelling language. Consequently, a meta model-based specification of the modelling language the patterns are defined for is necessary for the application of our approach. In this paper, we provide an application example for Event-driven Process Chains (EPC) [Sc00] (cf. Section 4). Furthermore, we present a prototypical modelling tool implementation that shows the applicability of the approach. The example of Section 4 serves again as the application scenario (cf. Section 5). Finally, we conclude our paper and outline further need for research (cf. Section 6).

2 Related Work

Supporting the structural analysis of conceptual models, fundamental work is done in the field of graph theory addressing the problem of graph pattern matching [GMS05; Fu95; VVS06, VM97]. Based on a given graph, these approaches discuss the identification of structurally equivalent (homomorphism) or synonymous (isomorphism) parts of the given graph in other graphs. To identify such parts, several pattern matching algorithms are proposed, which make use of a pattern definition as comparison criteria to find corresponding parts in other graphs. The algorithms compute walks through the graphs in order to analyze its nodes and its structure. As a result, they identify patterns representing corresponding parts of the compared graphs. Thus, a pattern is based on a particular labelled graph section and is not predefined independently. Some approaches are limited to specific types of graphs (e.g., the approaches of [Fu95; VaVS06] are restricted to labelled directed graphs).

In the context of process models, so-called behavioural approaches have been proposed [Hi93; MAW08; Hi05]. Two process models are considered equivalent if they behave identically during simulation. This implies that the respective modelling languages possess formal execution semantics. Therefore, the authors focus on Petri Nets and other workflow modelling languages [DDM08]. Moreover, due to the requirement of model simulation, these approaches generally consider process models as a whole. Patterns as model subsets are only comparable if they are also executable. Hence, not every pattern – even if provided with formal execution semantics – can be used for matching.

In the domain of database engineering, various approaches have been presented, which address the problem of schema matching. Two input schemas (i.e., descriptions of database structures) are taken and mappings between semantically corresponding elements are produced [RB01]. These approaches operate on single elements only [LC00] or assume that the schemas have a tree-like structure [MBR01]. Recently, the methods developed in the context of database schema matching have been applied in the field of ontology matching as well [Au05]. Additionally, approaches explicitly dedicated to matching ontologies have been presented. They usually utilize additional context information (e.g., a corresponding collection of documents [SM01]), which is not given in standard conceptual modeling settings. Moreover, as schema-matching approaches operate on approximation-basis, similar structures – and not exact pattern occurrences – are addressed. Consequently, these approaches lack the opportunity of including explicit structure descriptions (e.g., paths of a given length or loops not containing given elements) in the patterns.

Design patterns are used in systems analysis and design to describe best-practice solutions for common recurring problems. Common design situations are identified, which can be modelled in various ways. The most desirable solution is identified as a pattern and recommended for further usage. The general idea originates from [AIS77], who identified and described patterns in the field of architecture. [Ga95] and [Fo02] popularized this idea in the domain of object-oriented systems design. Workflow patterns, that is patterns applied to workflow models, is another dynamically developing research domain regarding patterns [Aa03]. However, the authors do not consider pattern matching. Instead, the modeller is expected to adopt the patterns as best-practice and to apply them intuitively whenever a common problem situation is met. A methodical pattern matching support is not addressed.

Patterns are also proposed as an indicator for possible conflicts typically occurring in the modelling and model integration process. [Ha94] proposes a set of general patterns for Entity-Relationship Models (ERMs [Ch76]). On the one hand, these patterns depict possible structural errors that may occur. For such error patterns corresponding patterns are proposed, which provide correct structures. On the other hand, sets of model patterns are discussed, which possibly lead to conflicts while integrating such models into a total model. Similar work in the field of process modelling is done by [Me07]. Based on the analysis of EPCs, he detects a set of general patterns, which depict syntactical errors in EPCs. However, these two approaches focus on particular structural patterns for specific modelling languages rather than a pattern definition and matching approach for arbitrary modelling languages.

3 Specification of Structural Model Patterns

3.1 Sets as a Basis for Pattern Matching

The idea of our approach is to regard a conceptual model as a set of model elements. Here, we further distinguish between objects representing nodes and relationships representing edges interrelating objects. Starting from this set, pattern matches are searched by performing set operations on this basic set. By combining different set operations, the pattern is built up successively. Given a pattern definition, the matching process returns a set of model subsets representing the pattern matches found. Every match found is put into an own subset. The following example illustrates the general idea.

A pattern definition consists of three objects of different types that are interrelated with each other by relationships. A pattern match within a model is represented as a set containing three different objects and three relationships that connect them. To distinguish multiple pattern matches, each match is represented as a separate subset. Thus, the result of a pattern matching process is represented by a set of pattern matches (i.e., a set of sets, cf. Fig. 1).

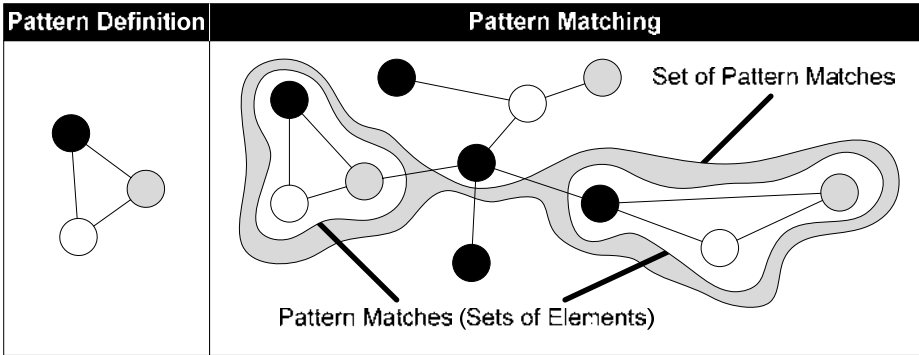


Fig. 1. Representation of Pattern Matches through Sets of Elements

3.2 Definition of Basic Sets

Therefore, as a basis for the specification of structural model patterns, we use a generic specification environment for conceptual modelling languages and models (cf. Fig. 2) applying the Entity-Relationship notation with (min,max)-cardinalities [ISO82]. Modelling languages typically consist of modelling objects that are interrelated through relationships (e.g., nodes and edges). In some modelling languages, relationships can be interrelated in turn (e.g., association classes in UML Class Diagrams [OMG09]).

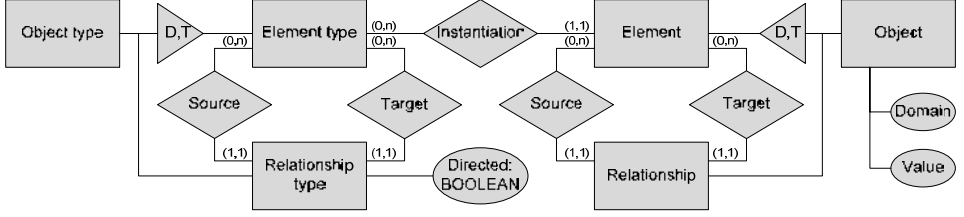


Fig. 2. Generic Specification Environment for Conceptual Modelling Languages and Models

Hence, modelling languages consist of *element types*, which are specialized into *object types* (i.e., nodes) and *relationship types* (e.g., edges and links). In order to allow relationships between relationships, the relationship type is defined as a specialization of the element type. Each relationship type has a *source* element type, from which it originates, and a *target* element type, to which it leads. Relationship types are either directed or undirected. Whenever the attribute *directed* is *FALSE*, the direction of the relationship type is ignored. The *instantiation* of modelling languages leads to models, which consist of particular *elements*. These are instantiated from their distinct element type. Elements are specialized into *objects* and *relationships*. Each of the latter leads from a *source element* to a *target element*. Objects can have *values* which are part of a distinct *domain*. For example, the value of an object “name” contains the string of the name (e.g., “product”). As a consequence, the domain of the object “name” has to be “string” in this case. Thus, attributes are considered as objects.

For the specification of structural model patterns we define the following sets, elements, and properties originating from the specification environment:

- E : set of all elements available; $e \in E$ is a particular element.
- $P(E)$: power set of E .
- O : set of all objects available; $O \subseteq E$; $o \in O$ is a particular object.
- R : set of all relationships available; $R \subseteq E$; $r \in R$ is a particular relationship.
- A : set of all element types available; $a \in A$ is a particular element type.
- B : set of all object types available; $B \subseteq A$; $b \in B$ is a particular object type.
- C : set of all relationship types available; $C \subseteq A$; $c \in C$ is a particular relationship type.
- I : set of all instantiations available; $I \subseteq A \times E$; $(a, e) \in I$ is a particular instantiation.
- T : set of all relationship targets available; $T \subseteq E \times R$; $(e, r) \in T$ is a particular target.
- S : set of all relationship sources available; $S \subseteq E \times R$; $(e, r) \in S$ is a particular source.
- X : set of elements with $x \in X \subseteq E$.
- X_k : sets of elements with $X_k \subseteq E$ and $k \in \mathbf{N}_0$
- x_l : distinct elements with $x_l \in E$ and $l \in \mathbf{N}_0$
- Y : set of objects with $y \in Y \subseteq O$.
- Z : set of relationships with $z \in Z \subseteq R$.
- $directed(c)$: property *directed* of a particular relationship type c .
- $domain(o)$: property *domain* of a particular object o .
- $value(o)$: property *value* of a particular object o .
- n_x : positive natural number $n_x \in \mathbf{N}_1$

- R_d : set of all directed relationships available;
 $R_d \subseteq R, ((c_d, r_d) \in I \wedge \text{directed}(c_d) = \text{TRUE} \wedge c_d \in C) \quad \forall r_d \in R_d$
- T_d : set of all directed relationship targets available; $T_d \subseteq T, (r_d \in R_d) \quad \forall (e, r_d) \in T$
- S_d : set of all directed relationship sources available; $S_d \subseteq S, (r_d \in R_d) \quad \forall (e, r_d) \in S$
- T_u and S_u are undirected counterparts; $T_u = T \setminus T_d$ and $S_u = S \setminus S_d$

3.3 Definition of Set-modifying Functions

Building up structural model patterns successively requires performing set operations on these basic sets. In the following, we introduce predefined functions on these sets in order to provide a convenient specification environment for structural model patterns dedicated to conceptual models. Each function has a defined number of input sets and returns a resulting set. For every function, we specify the input and output sets and provide a formal specification based on predicate logic. In addition, we provide textual explanations where necessary. First, since a goal of the approach is to specify any structural pattern, we must be able to reveal specific properties of model elements (e.g., type, value, or value domain):

- $\text{ElementsOfType}(X, a) \subseteq E$ is provided with a set of elements X and a distinct element type a . It returns a set containing all elements of X that belong to the given type:

$$\text{ElementsOfType}(X, a) = \{x \in X \mid (a, x) \in I\}$$

- $\text{ObjectsWithValue}(Y, \text{value}Y) \subseteq O$ takes a set of objects Y and a distinct value $\text{value}Y$. It returns a set containing all objects of Y whose values equal the given one:

$$\text{ObjectsWithValue}(Y, \text{value}Y) = \{y \in Y \mid \text{value}(y) = \text{value}Y\}$$

- $\text{ObjectsWithDomain}(Y, \text{domain}Y) \subseteq O$ takes a set of objects Y and a distinct domain $\text{domain}Y$. It returns a set with all objects of Y whose domains equal the given one:

$$\text{ObjectsWithDomain}(Y, \text{domain}Y) = \{y \in Y \mid \text{domain}(y) = \text{domain}Y\}$$

Second, relations between elements have to be revealed in order to assemble complex pattern structures successively. Functions are required that combine elements and their relationships and elements that are related respectively.

- $\text{ElementsWithRelations}(X, Z) \subseteq \mathcal{P}(E)$ is provided with a set of elements X and a set of relationships Z . It returns a set of sets containing all elements of X and all relationships of Z , which are connected. Each occurrence is represented by an inner set:

$$\text{EWR}(x_1, Z) = \{z \in Z \mid (x_1, z) \in T \vee (x_1, z) \in S\} \cup \{x_1\}, x_1 \in E$$

$$\text{ElementsWithRelations}(X, Z) = \{\text{EWR}(x, Z)\}, x \in X$$

- $\text{ElementsWithoutRelations}(X, Z_d) \subseteq \mathcal{P}(E)$ is provided with a set of elements X and a set of relationships Z . It returns a set of sets containing all elements of X that are connected to outgoing relationships of Z , including these relationships. Each occurrence is represented by an inner set:

$$\text{EWOR}(x_1, Z_d) = \{z_d \in Z_d \mid (x_1, z_d) \in S_d\} \cup \{x_1\}, x_1 \in E$$

$$\text{ElementsWithoutRelations}(X, Z_d) = \{\text{EWOR}(x, Z_d)\}, x \in X$$

path elements are handled analogously to *Paths*. Each loop found is represented by an inner set:

$$\text{Loops}(X) = \bigcup_{x \in X} PX(x, x)$$

- $\text{DirectedLoops}(X) \subseteq \mathbf{P}(E)$ is defined analogously:

$$\text{DirectedLoops}(X) = \bigcup_{x \in X} DPX(x, x)$$

In order to provide a convenient specification environment for structural model patterns, we define some additional functions that are derived from those already introduced:

- $\text{ElementsWithRelationsOfType}(X, Z_d, c_d) \subseteq \mathbf{P}(E)$ is provided with a set of elements X , a set of relationships Z_d and a distinct relationship type c_d . It returns a set of sets containing all elements of X and relationships of Z_d of the type c_d , which are connected. Each occurrence is represented by an inner set:

$$\begin{aligned} \text{ElementsWithRelationsOfType}(X, Z_d, c_d) = \\ \text{ElementsWithRelations}(X, \text{ElementsOfType}(Z_d, c_d)) \end{aligned}$$

- $\text{ElementsWithoutRelationsOfType}(X, Z_d, c_d) \subseteq \mathbf{P}(E)$ is provided with a set of elements X , a set of relationships Z_d and a distinct relationship type c_d . It returns a set of sets containing all elements of X that are connected to outgoing relationships of Z_d of the type c_d , including these relationships. Each occurrence is represented by an inner set:

$$\begin{aligned} \text{ElementsWithoutRelationsOfType}(X, Z_d, c_d) = \\ \text{ElementsWithoutRelations}(X, \text{ElementsOfType}(Z_d, c_d)) \end{aligned}$$

- $\text{ElementsWithInRelationsOfType}(X, Z_d, c_d) \subseteq \mathbf{P}(E)$ is defined analogously to $\text{ElementsWithoutRelationsOfType}$:

$$\begin{aligned} \text{ElementsWithInRelationsOfType}(X, Z_d, c_d) = \\ \text{ElementsWithInRelations}(X, \text{ElementsOfType}(Z_d, c_d)) \end{aligned}$$

- $\text{ElementsWithNumberOfRelations}(X, n_x) \subseteq \mathbf{P}(E)$ is provided with a set of elements X and a distinct number n_x . It returns a set of sets containing all elements of X , which are connected to the given number of relationships of R , including these relationships. Each occurrence is represented by an inner set:

$$\begin{aligned} \text{EWN}(x) = \{r \in R \mid (x, r) \in T \vee (x, r) \in S\} \cup \{x\} \\ \text{ElementsWithNumberOfRelations}(X, n_x) = \{\text{EWN}(x) \mid |\text{EWN}(x)| = n_x + 1\} \end{aligned}$$

- $\text{ElementsWithNumberOfOutRelations}(X, n_x) \subseteq \mathbf{P}(E)$ and $\text{ElementsWithNumberOfInRelations}(X, n_x) \subseteq \mathbf{P}(E)$ are defined analogously:

$$\begin{aligned} \text{EWNIR}(x) = \{r \in R_d \mid (x, r) \in T_d\} \cup \{x\} \\ \text{ElementsWithNumberOfInRelations}(X, n_x) = \{\text{EWNIR}(x) \mid |\text{EWNIR}(x)| = n_x + 1\} \end{aligned}$$

$$\begin{aligned} \text{EWNOR}(x) = \{r \in R_d \mid (x, r) \in S_d\} \cup \{x\} \\ \text{ElementsWithNumberOfOutRelations}(X, n_x) = \{\text{EWNOR}(x) \mid |\text{EWNOR}(x)| = n_x + 1\} \end{aligned}$$

- $\text{ElementsWithNumberOfRelationsOfType}(X, c, n_x) \subseteq \mathbf{P}(E)$ is provided with a set of elements X , a distinct relationship type c and a distinct number n_x . It returns a set of sets containing all elements of X , which are connected to the given number of relationships of R of the type c , including these relationships. Each occurrence is represented by an inner set:

$EWNRT(x, c) = \{r \in R \mid (c, r) \in I \wedge (x, r) \in T \vee (x, r) \in S\} \cup \{x\}$
 $ElementsWithNumberOfRelationsOfType(X, c, n_x) =$
 $\{EWNRT(x, c) \mid |EWNRT(x, c)| = n_x + 1\}$

- $ElementsWithNumberOfOutRelationsOfType(X, c_d, n_x) \subseteq \mathbf{P}(E)$ and $ElementsWithNumberOfInRelationsOfType(X, c_d, n_x) \subseteq \mathbf{P}(E)$ are defined analogously:

$\circ EWNIRT(x, c_d) = \{r \in R_d \mid (c_d, r) \in I \wedge (x, r) \in T_d\} \cup \{x\}$
 $ElementsWithNumberOfInRelationsOfType(X, c_d, n_x) =$
 $\{EWNIRT(x, c_d) \mid |EWNIRT(x, c_d)| = n_x + 1\}$

$\circ EWNORT(x, c_d) = \{r \in R_d \mid (c_d, r) \in I \wedge (x, r) \in S_d\} \cup \{x\}$
 $ElementsWithNumberOfOutRelationsOfType(X, c_d, n_x) =$
 $\{EWNORT(x, c_d) \mid |EWNORT(x, c_d)| = n_x + 1\}$

- $PathsContainingElements(X_l, X_n, X_c) \subseteq \mathbf{P}(E)$ is provided with three sets of elements X_l, X_n and X_c . It returns a set of sets containing elements that represent all paths from elements of X_l to elements of X_n , which each contain at least one element of X_c . The direction of relations and path elements are handled analogously to $Paths$. Each path found is represented by an inner set:

$PCE(x_l, x_n, X_c) = \{Set((x_1, x_2, \dots, x_n)) \mid x_2, \dots, x_{n-1} \in E \wedge \exists x_c \in \{x_2, \dots, x_{n-1}\} \wedge$
 $((x_i, x_{i+1}) \in S_u \vee (x_i, x_{i+1}) \in T_u) \forall 1 \leq i < n\}$
 $PathsContainingElements(X_l, X_n, X_c) = \bigcup_{x_l \in X_l, x_n \in X_n} PCE(x_l, x_n, X_c)$

- $DirectedPathsContainingElements(X_l, X_n, X_c) \subseteq \mathbf{P}(E)$, $PathsNotContainingElements(X_l, X_n, X_c) \subseteq \mathbf{P}(E)$, and $DirectedPathsNotContainingElements(X_l, X_n, X_c) \subseteq \mathbf{P}(E)$ are defined analogously:

$\circ DPCE(x_l, x_n, X_c) = \{Set((x_1, x_2, \dots, x_n)) \mid x_2, \dots, x_{n-1} \in E \wedge \exists x_c \in \{x_2, \dots, x_{n-1}\} \wedge$
 $((x_{2i-1}, x_{2i}) \in S_d \wedge (x_{2i+1}, x_{2i}) \in T_d \forall 1 \leq i \leq \lfloor n/2 \rfloor)$
 $\vee ((x_{2i}, x_{2i-1}) \in T_d \wedge (x_{2i}, x_{2i+1}) \in S_d \forall 1 \leq i \leq \lfloor n/2 \rfloor)$
 $\vee ((x_{2i-1}, x_{2i}) \in S_d \wedge (x_{2i+1}, x_{2i}) \in T_d \wedge (x_{n-1}, x_n) \in S_d \forall 1 \leq i \leq \lfloor n/2 \rfloor - 1)$
 $\vee ((x_{2i}, x_{2i-1}) \in T_d \wedge (x_{2i}, x_{2i+1}) \in S_d \wedge (x_n, x_{n-1}) \in T_d \forall 1 \leq i \leq \lfloor n/2 \rfloor - 1) \forall 1 \leq i < n\}$
 $DirectedPathsContainingElements(X_l, X_n, X_c) = \bigcup_{x_l \in X_l, x_n \in X_n} DPCE(x_l, x_n, X_c)$

$\circ PNCE(x_l, x_n, X_c) = \{Set((x_1, x_2, \dots, x_n)) \mid x_2, \dots, x_{n-1}$
 $\in E \setminus X_c \wedge ((x_i, x_{i+1}) \in S_u \vee (x_i, x_{i+1}) \in T_u) \forall 1 \leq i < n\}$
 $PathsNotContainingElements(X_l, X_n, X_c) = \bigcup_{x_l \in X_l, x_n \in X_n} PNCE(x_l, x_n, X_c)$

$\circ DPNCE(x_l, x_n, X_c) = \{Set((x_1, x_2, \dots, x_n)) \mid x_2, \dots, x_{n-1} \in E \setminus X_c$
 $\wedge (((x_{2i-1}, x_{2i}) \in S_d \wedge (x_{2i+1}, x_{2i}) \in T_d \vee 1 \leq i \leq \lfloor n/2 \rfloor)$
 $\vee ((x_{2i}, x_{2i-1}) \in T_d \wedge (x_{2i}, x_{2i+1}) \in S_d \vee 1 \leq i \leq \lfloor n/2 \rfloor)$
 $\vee ((x_{2i-1}, x_{2i}) \in S_d \wedge (x_{2i+1}, x_{2i}) \in T_d \wedge (x_{n-1}, x_n) \in S_d \vee 1 \leq i \leq \lfloor n/2 \rfloor - 1)$
 $\vee ((x_{2i}, x_{2i-1}) \in T_d \wedge (x_{2i}, x_{2i+1}) \in S_d \wedge (x_n, x_{n-1}) \in T_d \vee 1 \leq i \leq \lfloor n/2 \rfloor - 1) \vee 1 \leq i < n\}$
 $DirectedPathsNotContainingElements(X_l, X_n, X_c) = \bigcup_{x_l \in X_l, x_n \in X_n} DPNCE(x_l, x_n, X_c)$

- $LoopsContainingElements(X, X_c) \subseteq \mathbf{P}(E)$, $DirectedLoopsContainingElements(X, X_c) \subseteq \mathbf{P}(E)$, $LoopsNotContainingElements(X, X_c) \subseteq \mathbf{P}(E)$, and $DirectedLoopsNotContainingElements(X, X_c) \subseteq \mathbf{P}(E)$ are defined analogously:

$\circ LoopsContainingElements(X, X_c) = \bigcup_{x \in X} PCE(x, x, X_c)$

- $DirectedLoopsContainingElements(X, X_c) = \bigcup_{x \in X} DPCE(x, x, X_c)$
- $LoopsNotContainingElements(X, X_c) = \bigcup_{x \in X} PNCE(x, x, X_c)$
- $DirectedLoopsNotContainingElements(X, X_c) = \bigcup_{x \in X} DPNCE(x, x, X_c)$

3.4 Definition of Set Operators for Sets of Sets

By nesting the functions introduced above, it is possible to build up structural model patterns successively. The results of each function can be reused adopting them as an input for other functions. In order to combine different results, the basic set operators *union* (\cup), *intersection* (\cap), and *complement* (\setminus) can be used generally. Since it should be possible to combine not only sets of pattern matches (i.e., sets of sets) but also the pattern matches themselves, this is the inner sets, we define additional set operators. These operate on the inner sets of two sets of sets respectively (cf. Table 1).

| Basic Sets | Operator Definition | Operator Symbol |
|--|---|-----------------|
| $F, G \subseteq \mathbf{P}(E), f \in F, g \in G$ | $Join(F, G) = \{f \cup g \mid \exists e \in E: e \in f \wedge e \in g\}$ | $F \cup G$ |
| $F, G \subseteq \mathbf{P}(E), f \in F, g \in G$ | $InnerIntersection(F, G) = \{f \cap g\}$ | $F \cap G$ |
| $F, G \subseteq \mathbf{P}(E), f \in F, g \in G$ | $InnerComplement(F, G) = \{f \setminus g \mid \exists e \in E: e \in f \wedge e \notin g\}$ | $F \setminus G$ |
| $F \subseteq \mathbf{P}(E), f \in F$ | $SelfUnion(F) = \bigcup_{f \in F} f$ | $\bigcup F$ |
| $F \subseteq \mathbf{P}(E), f \in F$ | $SelfIntersection(F) = \bigcap_{f \in F} f$ | $\bigcap F$ |

Table 1. Set Operators for Sets of Sets

The *Join* operator performs a *Union* operation on each inner set of the first set with each inner set of the second set. Since we regard patterns as cohesive, only inner sets that have at least one element in common are considered. The *InnerIntersection* operator *intersects* each inner set of the first set with each inner set of the second set. The *InnerComplement* operator applies a *complement* operation to each inner set of the first outer set combined with each inner set of the second outer set. Only inner sets that have at least one element in common are considered.

As most of the functions introduced in Section 3.3 expect simple sets of elements as inputs, we introduce further operators that turn sets of sets into simple sets. The *SelfUnion* operator merges all inner sets of one set of sets into a single set performing a *union* operation on all inner sets. The *SelfIntersection* operator performs an *intersection* operation on all inner sets of a set of sets successively. The result is a set containing elements that each occur in all inner sets of the original outer set.

4 Application of Structural Model Patterns

To illustrate the usage of the set functions, we apply our approach to an EPC application scenario. In the scenario, the approach is applied to complex syntax verification in EPCs.

Therefore, we regard a simplified modelling language of EPCs. Models of this language consist of the object types function, event, AND connector, OR connector, and XOR connector (i.e., $B = \{\text{function, event, AND, OR, XOR}\}$). Furthermore, EPCs consist of different relationship types that lead from any object type to any other object type, except from function to function and from event to event. All these relationship types are directed, (i.e., $c.directed = TRUE \ \forall c \in C$).

A common error in EPCs is that decisions (i.e., XOR or OR splits) are modelled successively to an event. Since events are passive element types of an EPC, they are not able to make a decision [Sc00]. Hence, any directed path in an EPC that reaches from an event to a function and contains no further events or functions but an XOR or OR split is a syntax error. In order to reveal such errors, we specify the following exemplary structural model pattern:

| | |
|---|---|
| DirectedPathsNotContainingElements (ElementsOfType (O, 'Event'), ElementsOfType (O, 'Function'), (ElementsOfType (O, 'Event') UNION ElementsOfType (O, 'Function'))) INTERSECTION | 1 |
| DirectedPathsContainingElements (ElementsOfType (O, 'Event'), ElementsOfType (O, 'Function'), | 2 |
| ((ElementsOfType (O, 'OR') UNION ElementsOfType(O, 'XOR')) COMPLEMENT | 3 |
| (O INNER_INTERSECTION (ElementsWithNumberOfOutRelations ((ElementsOfType (O, 'XOR') UNION ElementsOfType (O, 'OR')), 1) UNION ElementsWithNumberOfOutRelations ((ElementsOfType (O, 'XOR') UNION ElementsOfType (O, 'OR')), 0))))) | 4 |

The first expression (cf. 1st block) determines all paths that start with an event and end with a function and do not contain any further functions or events. The result is intersected with all paths starting with an event and ending with a function (cf. 2nd block) that contain OR and/or XOR connectors (cf. 3rd block), but only those that are connected to 2 or more outgoing relationships. Thus, these XORs and ORs are subtracted by XORs and ORs that are only connected to one or less relationship(s) (cf. 4th block). Summarizing, all paths are returned that lead from an event to a function not containing any further events and functions, and that contain splitting XOR and/or OR connectors (cf. Section 5 for implementation issues and exemplary results). This way, any syntax error pattern can be specified and applied to any model base.

5 Tool Support

In order to show the feasibility of the approach, we have implemented a plug-in for a meta modelling tool that was available from a former research project. The tool consists of a meta modelling environment that is based on the generic specification approach for modelling languages shown in Fig. 2.

The plug-in provides a specification environment for structural model patterns, which is integrated into the meta modelling environment of the tool, since the patterns are dependent on the respective modelling language. All basic sets, functions, and set operators introduced in Section 3 are provided and can be used to build up structural model patterns successively. In order to gain a better overview over the patterns, they are displayed and edited in a tree structure. The tree-structure is built up through drag-and-drop of the basic sets, functions and set operators. Whenever special characteristics of an according modelling language (function, event etc.) or variables such as numeric values or names are used for the specification, this is expressed by using a “variable” element. The variable element, in turn, is instantiated by selecting a language-specific characteristic from a menu or by entering a particular value (such as “2”).

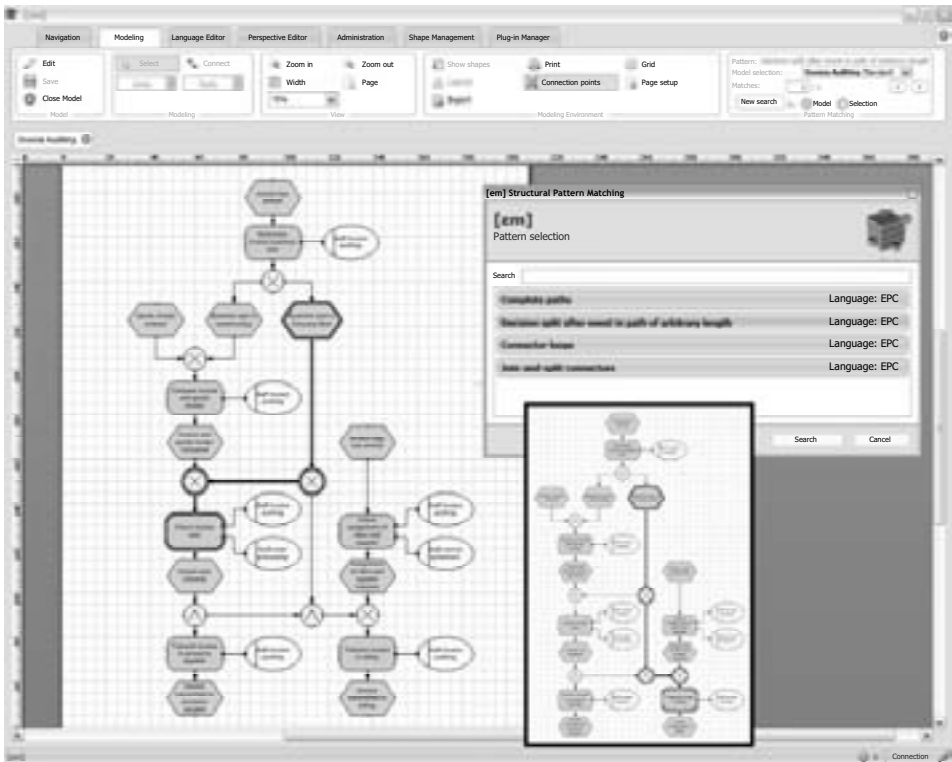


Fig. 3. Result of the Pattern Matching Process of “Decision split after event...”

The patterns specified can be applied to any model that is available within the model base and that was developed with the according modelling language. Fig. 3 shows an exemplary model that was developed with the modelling language of EPCs and that contains a syntax error consisting of a decision split following an event. The structural model pattern matching process is started by selecting the appropriate pattern to search for. Every match found is displayed by marking the according model section. The user can switch between different matches. In our example, two matches are found, as the

decision split following the event leads to two different paths (the second match is shown in the lower right corner of Fig. 3).

6 Conclusion and Outlook

Supporting model analysis by a generic pattern matching approach is promising, as it is not restricted to a particular problem area or modelling language. A first rudimentary evaluation through implementation and exemplary application of the approach has shown its general feasibility. Nevertheless, there still remains need for further research.

In the short term, we will focus on completing the evaluation of the presented approach. Although our current prototypical implementation already shows its general feasibility, further evaluation of our approach is necessary. We will conduct a series of with-without experiments in real-world scenarios. They will show if the presented function set is complete, if the ease of use is satisfactory for users not involved in the development of the approach, and if the application of the approach actually leads to an improved model analysis support. Although we strongly believe that our tool-implemented approach will inevitably support modellers in the task of model analysis and integration, this needs to be objectively proven.

Medium-term research will address further applications for the structural model pattern matching approach presented here. For instance, we will question if modelling conventions on the basis of structural model patterns that are provided prior to modelling are able to increase the comparability of conceptual models.

References

- [Au05] Aumuellner, D., Do, H.-H., Massmann, S., Rahm, E.: Schema and ontology matching with COMA++. In: Proceedings of the 2005 ACM SIGMOD international Conference on Management of Data, New York, 2005; pp. 906-908.
- [Aa03] van der Aalst, W. M. P.; ter Hofstede, A. H. M.; Kiepuszewski, B.; Barros, A. P.: Workflow Patterns. In: Distributed and Parallel Databases 14 (2003) 3; pp. 5-51.
- [AIS77] Alexander, C.; Ishikawa, S.; Silverstein, M. A.: Pattern Language. New York, 1977.
- [Ch76] Chen, P.P.-S.: The Entity-Relationship Model: Toward a Unified View of Data. In: ACM Transactions on Database Systems 1 (1976) 1; pp. 9-36.
- [DDM08] van Dongen, B. F.; Dijkman, R.; Mendling, J.: Measuring similarity between business process models. In (Bellahsene, Z.; Léonard, M. eds.): Proceedings of the 20th International Conference on Advanced Information Systems Engineering, Montpellier, 2008; pp. 450-464.
- [Fo02] Fowler, M.: Patterns of Enterprise Application Architecture. Reading, 2002.
- [Fu95] Fu, J.: Pattern matching in directed graphs. In (Galil, Z.; Ukkonen, E. eds.): Proceedings of the 6th Annual Symposium on Combinatorial Pattern Matching, Espoo, 1995; pp. 64-77.
- [Ga95] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Design Patterns - Elements of Reusable Object-Oriented Software. New York, 1995.

- [GMS05] Gori, M.; Maggini M.; Sarti, L.: The RW2 algorithm for exact graph matching. In (Singh, S.; Singh, M.; Apté, C.; Perner, P. eds.): Proceedings of the 4th International Conference on Advances in Pattern Recognition, Bath, 2005; pp. 81-88.
- [Ha94] Hars, A.: Reference Data Models - Foundations of Efficient Data Modeling [In German: Referenzdatenmodelle. Grundlagen effizienter Datenmodellierung]. Wiesbaden, 1994.
- [Hi05] Hidders, J.; Dumas, M.; van der Aalst, W. M. P.; ter Hofstede, A. H. M.; Verelst, J.: When are two workflows the same? In (Atkinson, M.; Dehne, F. eds.): Proceedings of the 11th Australasian Symposium on Theory of Computing, Newcastle, 2005; pp. 3-11.
- [Hi93] Hirschfeld, Y.: Petri nets and the equivalence problem. In (Börger, E.; Gurevich Y.; Meinke, K. eds.): Proceedings of the 7th Workshop on Computer Science Logic, Swansea, 1993; pp. 165-174.
- [ISO82] ISO: Concepts and Terminology for the conceptual Schema and the Information Base. Technical report ISO/TC97/SC5/WG3, 1982.
- [LC00] Li, W.; Clifton, C.: SemInt: a tool for identifying attribute correspondences in heterogeneous databases using neural network. In: Data & Knowledge Engineering 33 (2000) 1; pp. 49-84.
- [MBR01] Madhavan, J.; Bernstein, P. A.; Rahm, E.: Generic schema matching with Cupid. In (Apers, P. M. G.; Atzeni, P.; Ceri, S.; Paraboschi, S.; Ramamohanarao, K.; Snodgrass, R. T. eds.): Proceedings of the 27th International Conference on Very Large Data Bases, Rome, 2001; pp. 49-58.
- [MAW08] de Medeiros, A. K. A.; van der Aalst, W. M. P.; Weijters, A. J. M. M.: Quantifying process equivalence based on observed behavior. In: Data & Knowledge Engineering 64 (2008) 1; pp. 55-74.
- [Me07] Mendling, J.: Detection and Prediction of Errors in EPC Business Process Models. Doctoral Thesis, Vienna University of Economics and Business Administration. Vienna, 2007.
- [OMG09] Object Management Group (OMG): Unified Modeling Language (OMG UML), Infrastructure, V2.1.2, <http://www.omg.org/docs/formal/07-11-04.pdf>, 2009.
- [RB01] Rahm, E.; Bernstein, P. A.: A Survey of approaches to automatic schema matching. In: The VLDB Journal – The International Journal on Very Large Data Bases 10 (2001) 4; pp. 334-350.
- [Sc00] Scheer, A.-W.: ARIS – Business Process Modelling. 3rd Edition. Berlin et al., 2000.
- [SM01] Stumme, G., Mädche, A.: FCA-Merge: Bottom-up merging of ontologies. In (Nebel, B. ed.): Proceedings of the 17th International Joint Conference on Artificial Intelligence, Seattle, 2001; pp. 225-230.
- [VM97] Valiente, G.; Martínez, C.: An Algorithm for Graph Pattern-Matching. In (Baeza-Yates, R.; Ziviani, N. eds.): Proceedings of the 4nd South American Workshop on String Processing, Brighton, 2006; pp. 180-197.
- [VVS06] Varró, G.; Varró, D.; Schürr, A.: Incremental Graph Pattern Matching - Data Structure and Initial Experiments. In (Margaria, T.; Padberg, J.; Taentzer, G. eds.): Proceedings of the 2nd International Workshop on Graph and Model Transformation, Brighton, 2006.
- [VTM08] Vergidis, K.; Tiwari, A.; Majeed, B.: Business process analysis and optimization: beyond reengineering. In: IEEE Transactions on Systems, Man, and Cybernetics 38 (2008) 1; pp. 69-82.

Integration of Object Oriented Domain Modeling and Meta-Modeling

Antoine Schlechter, Guy Simon, Fernand Feltz

Département Informatique, Systèmes et Collaboration (ISC)
Centre de Recherche Public Gabriel Lippmann
41, rue du Brill
L-4422 Belvaux
schlecht@lippmann.lu
simon@lippmann.lu
feltz@lippmann.lu

Abstract: Despite a broad agreement on the benefits of model driven approaches to software engineering, the use of such techniques is still not very widespread. One of the major reasons is the appearing discouraging difficulty of meta-modeling. This paper illustrates the relations, dependencies and differences between a traditional abstract object-oriented domain model and a meta-model for the same domain. It presents a new approach to model driven engineering of enterprise application software that integrates domain modeling and meta-modeling in order to take full advantage of both traditional and generative software development methods.

1 Introduction

Since quite a while, model-driven approaches to software engineering such as Model Driven Engineering (MDE), Model Driven Software Development (MDSD) or Model Driven Architecture (MDA) have been advertised to be the solution to the ever-increasing complexity in software development. These techniques offer an easy way to domain-specific abstraction and to a high degree of automation in the coding process. Abstraction and automation lead to higher productivity, easier extensibility and better quality of the software.

Although there are success stories about MDA, MDSD and MDE, the adoption of such techniques in industry is not yet very widespread. [AK03], [Kü06] and [He06] see the reasons in a still incomplete and not yet fully understood theoretical foundation of MDE. Other researchers such as [PD08], [Sel08], [RR08], and [MFM08] investigated this issue from a more practical point of view and identified mainly two kinds of reasons. On the one hand we find so called technical reasons like bad tool support, missing tool documentation, insufficient interoperability between tools, lack of user-friendliness, and

others. On the other hand, a lot of programmers simply feel comfortable with their proven methods of software development. They often only see the discomfort, the difficulty, the threats and dangers but not the benefits in new technology. This shortcoming of awareness, education, and training is often referred to as cultural problems.

Although not all of the tool requirements from [Ke02] are fully achieved, there are tool chains such as EMF, GMF and oaw [Ecl09] that provide most of the needed functions for MDE at least for smaller-scale projects. In fact, [TG08] finds in a survey among several SMEs that the importance of tool support is “surprisingly low” when it comes to suggest improvements to current practices, whereas “methodology”, “increased awareness” and “training” are all mentioned significantly more often.

We are convinced, that the most important obstacle to the adoption of MDE is the appearing discouraging difficulty of meta-modeling, that is due to the lack of methods about how to address the specification of a meta-model or domain specific language (DSL) at the center of each model-driven approach.

In fact, there are papers that present special meta-models or domain specific languages [KK03], (references in [DKV00]). Besides, [LKT04], [MHS05], and [DKV00] identify several high level possibilities to define a meta-model or a DSL. Unfortunately, it remains unclear how to effectively bridge the gap between the domain analysis and the explicit definition of the meta-model or DSL.

[RJ96] proposes domain specific languages on top of a framework in order to make this framework more comfortable to use. [AC06] analyses the possibilities to design Framework Specific Modeling Languages (FSML) with roundtrip engineering. [Sa07] introduces annotations to the implementation of a framework that may be used to generate a DSL for this framework. These DSLs remain very close to the technical details of a given framework. They are defined on top of existing finished frameworks.

Since we would like to take the benefits of model driven engineering during the whole development process, we present an approach that integrates long-established object-oriented domain modeling and meta-modeling. First we define a traditional abstract object-oriented domain model. The idea is to generate the concrete subclasses with their supporting code from a so-called domain specific model. In order to formally describe such a model, we need to define a meta-model. A first version of this meta-model can be derived from the abstract domain model. After that, we may choose for each more advanced feature of the system whether to include it in the object-oriented domain model or in the meta-model. As experienced object-oriented software developers should feel comfortable building domain models following for instance the principles of Domain Driven Design (DDD)[Ev04], meta-modeling, the first, most important, most difficult and most discouraging step in model driven software development, should become easier for them just by following the concrete guidance from our method.

To present and illustrate our method, we will use the development of a Manufacturing Execution System (MES) as an example. First of all we introduce the “ubiquitous language” (from DDD) for MES, its representation as an object-oriented domain model and the software system architecture (section 2). Based on the architectural description we

outline a model-driven approach and explain its benefits (section 3). The meta-model at the center of the approach will be defined based on the object-oriented domain model (section 4). Both the domain-model and meta-model are an integral part of the proposed MDE approach and as such, they may be extended separately to take full advantage of both traditional and generative software development methods (section 5). The proposed method has been use to develop a concrete manufacturing execution system for a plastics injection molding company (section 6).

2 Object-Oriented Domain Model

Manufacturing Execution Systems deal with the collection, evaluation, analysis, interpretation and visualization of data from production in order to better control the production processes. The central objects of interest in MES are jobs. A job produces a product. Products have resource requirements used to determine what resources must be assigned to a job for the production of a given product. Jobs use time on resources, have an internal state and may contain several sets of values, so called variables, to represent data from quality control and process monitoring for instance. Input events may change the state and the variables of jobs. The history of a job's state is stored in a series of slots. Of course, all these objects may have attributes. Besides these domain specific objects, there are simple persistent data entities.

For the sake of simplicity, we will not go into detail for all of these aspects. A first extract from the abstract domain model for MES containing only entities, jobs, variables and input events is depicted at the top of Figure 1.

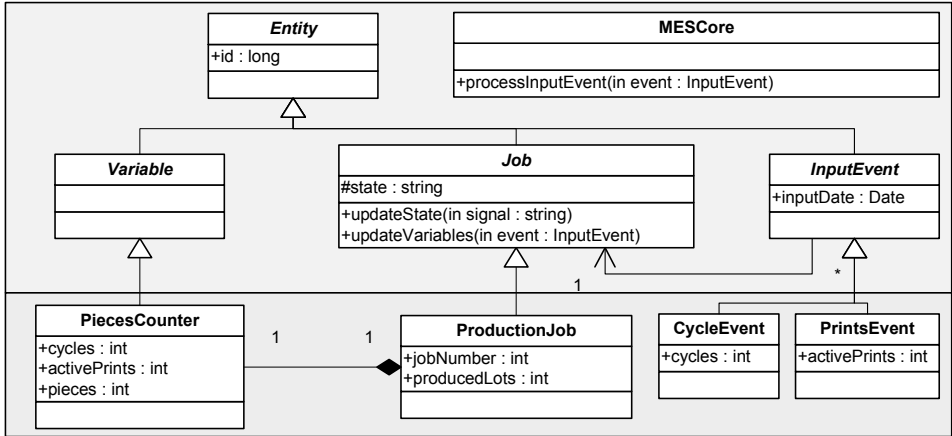


Figure 1: Class diagram with an extract of the abstract framework classes (upper part) and some concrete example subclasses for a simple MES (lower part).

This UML model represents the static abstract class structure of an MES implementation. It contains the data structure and some very basic operations and services. The `processInputEvent()` service of the **MESCore** class coordinates the different operations

on a job, its state, variables and history involved in the processing of an input event. For the implementation of a concrete MES, these abstract classes have to be specialized.

As an example, we assume that there is only one kind of job called `ProductionJob` that contains exactly one variable `PiecesCounter` used to count the produced pieces. In order to calculate the produced pieces, we need to know the number of currently active prints in a mold (=pieces produced per cycle). Additionally, we need input events to change the number of active prints (`PrintsEvent`) and to enter a number of cycles (`CycleEvent`). The respective classes are represented at the bottom of Figure 1.

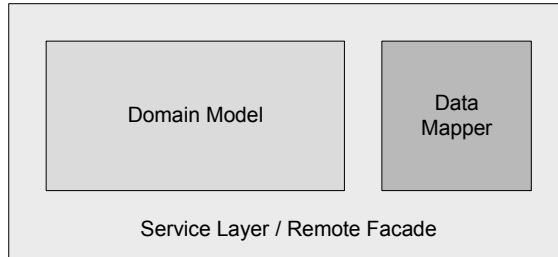


Figure 2: Architecture of the MES-Backend with a ‘Domain Model’ and a ‘Data Mapper’ on the inner layer and a ‘Service Layer’ implemented as a ‘Remote Façade’ on the outer layer. (Patterns from [Fo03])

As manufacturing execution systems are typically installed between already existing IT-Systems at the customer’s factory, the attributes of these concrete subclasses should be defined to be compatible with the data from the existing systems.

```

ProductionJob createProductionJob(int jobNumber, int producedLots)
ProductionJob getProductionJobByJobNumber(int jobNumber)
Collection<ProductionJob> getProductionJobsByState(String state)
void
    createPrintsEventForProductionJobByJobNumber(int jobNumber, int prints)
Collection<PrintsEvent>
    getPrintsEventsByProductionJobJobNumber(int jobNumber)
Collection<PrintsEvent>
    getPrintsEventsByProductionJobJobNumber(int jobNumber, Interval p)

```

Figure 3: Some services offered by the `JobManager` and the `InputEventManager` in the remote façade / service layer.

In order to keep the implementation of the interfaces between the MES and the surrounding software as simple as possible, we propose a layered architecture with a service layer / remote façade that exposes useful functions to remote and local clients as an outer layer. The services in the remote façade / service layer coordinate the access to domain objects in persistent storage via a data mapper with the necessary calls to services and object methods from the domain model (Figure 2). For a detailed explanation of the patterns remote façade, service layer, data mapper and domain model, we recommend [Fo03].

The service layer provides a `JobManager` and an `InputEventManager` with services for the creation and retrieval of jobs and input events respectively. Additionally, the creation

services for input events also process the created events by calling the MES-Core processInputEvent() service. These managers offer among others the services listed in Figure 3.

This short list of services contains quite a lot of redundancy when compared to the class diagram in Figure 1. Most of these services follow very strict and simple patterns depending on the abstract super-class, the name and attributes of the classes and possible relations between classes. At every modification we need to keep them consistent with the data structures. Thus, the implementation and maintenance of these services is a time-consuming, very repetitive and error-prone task. In fact, most of these services can completely be generated with a little more information than provided by a standard UML class diagram.

3 Domain Specific Model

Universal modeling languages like the UML are well suited to describe a given software solution. Often, the models are simply abstract representations of the code of an application, thus usually leaving out functional details or spreading these details over a lot of different types of models. Domain specific modeling languages are designed to capture the essence of a domain and the respective models are abstractions of the real world problem to be solved. From such models, we can generate all the abstract models of a solution. In addition, it is often possible to generate some functional details or even a complete application. For illustration, we will now present a possible domain specific model for our example MES-System.

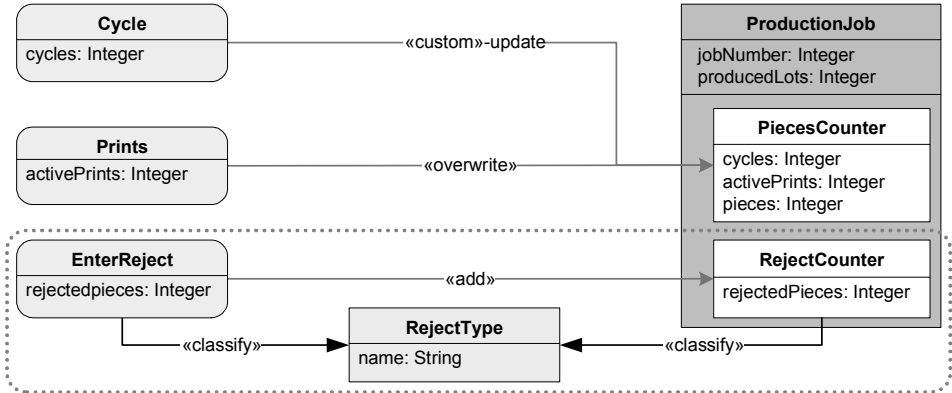


Figure 4: Extract from an MES model. The non-dotted area corresponds to the object model in the lower part of Figure 1; the dotted area represents the object model in Figure 5.

As seen in the analysis of the services needed in the remote façade, we need a distinction between the different class hierarchies in Figure 1. Thus, we model the subclasses using different representations. In Figure 4 subclasses of the InputEvent class are drawn as green rounded rectangles (left-hand side), direct subclasses of the Entity class are represented by gray rectangles (bottom), and subclasses of the Job class are modeled as blue

rectangles (right-hand side) containing a white rectangle for each owned subclass of the Variable class. The attributes of the classes are modeled just like in UML within the class representing shapes.

In addition to Figure 1, we added relationships between input events and variables that indicate if at all and how a given event updates a variable. In our example, the Cycle event will update the PiecesCounter variable in a custom way to be further specified manually in the generated code. The Prints event will overwrite the activePrints attribute of the PiecesCounter variable with the value of its attribute. Within the dotted area of Figure 4, we added an EnterReject event and a RejectCounter variable that both use RejectType to classify their attribute rejectedPieces. The corresponding data structures are given in Figure 5. In fact, the RejectCounter variable will have a hashtable to map reject types to the corresponding total number of rejected pieces.

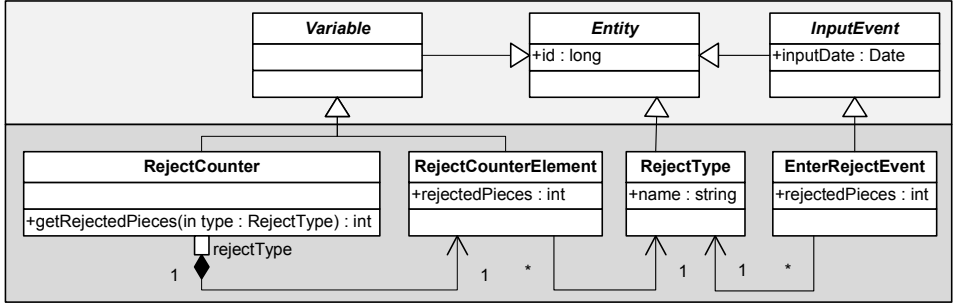


Figure 5: Class diagram corresponding to the dotted area in the MES model of Figure 4.

For now, Figure 4 is just a picture and we need to define its formal semantics in order to make it a model [KI07]. We have drawn this new model with clear “translational semantics” in mind. In fact, we constructed its elements directly from the subclasses of our abstract domain model they represent. It should be clear, that the models in Figure 1 and Figure 5 as well as the services in Figure 3 can be generated from the model in Figure 4.

In this new model, we left out technical details and added domain specific details like the update relationships between input events and variables. This additional information can be used to generate the major part of the `updateVariables()` method of the `ProductionJob` class together with the needed supporting methods within the different input events. Besides, the new model is more readable and has an intuitive meaning that is easily understandable by domain experts.

4 From Domain Modeling to Meta-Modeling

After the presentation of a possible domain specific model and its advantages, we need to formally define the corresponding meta-model. Despite the higher effort for tool building involved with a real new meta-model, we chose this way because of its other advantages over an UML profile or an UML extension [WS07].

As a first step in the definition of the meta-model, we basically just take the classes of our abstract object oriented domain model in the upper part of Figure 1 and put them as meta-classes in the meta-model in Figure 6. After this first step, we find the meta-classes JobClass, VariableClass, InputEventClass and EntityClass in our meta-model. Instances of these meta-classes represent subclasses of the domain classes Job, Variable, InputEvent and Entity. For instance, the blue rectangle representing the ProductionJob in Figure 4 is an instance of the meta-class JobClass.

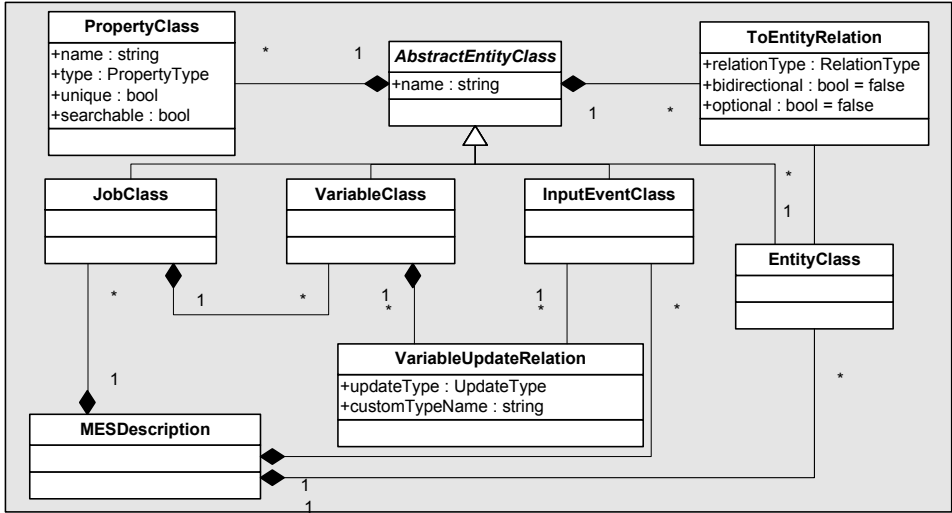


Figure 6: Extract from the MES meta-model.

All these instances should have a name (“ProductionJob”, “RejectCounter”, “PrintsEvent”, “RejectType”, ...). It should be possible, to define attributes for these instances (jobNumber, rejectedPieces, ...). To capture these commonalities in a central element, we introduce the common super-(meta-)class AbstractEntityClass in our meta-model. It has a string-type attribute name to take the names of the instances. AbstractEntityClasses may have several PropertyClasses to represent the attributes of their instances. For instance, ProductionJob is an instance of JobClass, that is an AbstractEntityClass with name=“ProductionJob”. Its attribute jobNumber with type “Integer” is an instance of PropertyClass with name=“jobNumber” and type=PropertyType::Integer.

The additional attributes unique and searchable of PropertyClass are used to indicate whether an attribute of a domain object (=instance of PropertyClass) can be used as an identifier for this domain object and whether it can be used to lookup and retrieve instances of this domain object. They are used to control which accessing services should be generated. The service getProductionJobByJobNumber() (Figure 3) for instance is generated because the attribute jobNumber of ProductionJob is marked to be unique and searchable.

Besides attributes (instances of PropertyClass), we would like to associate simple data entities to our model elements (instances of JobClass, ...) in order to build normalized

data structures. Each such association in a domain specific model is an instance of the ToEntityRelation meta-class. The attributes of this meta-class are used to control the type (classifying or not, cardinalities) and other options of the associations.

Variations of this AbstractEntityClass-pattern will be a part of practically every meta-model that is used to model traditional data structures. Another returning aspect is the need for models to have one single point of entry. Therefore all top-level elements in a model (those that are not owned by other model elements) must be owned by one object of a special type. In our case this is MESDescription.

The only substantial additions to the meta-model are the composition relationship between JobClass and VariableClass indicating that jobs may contain variables and the VariableUpdateRelation meta-class to model which input events update which variables and how.

5 Integrated Domain Modeling and Meta-Modeling

Our approach contains two separate development processes: the first one is about the development of applications; the second one is about the development of tools that are used in application development (Figure 7).

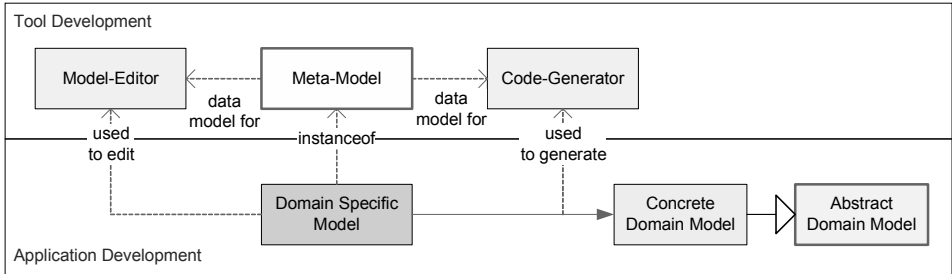


Figure 7: Overview of the integrated domain modeling and meta-modeling approach.

We start the application development traditionally with an abstract object-oriented domain model. Instead of manually extending and specializing this abstract domain model into a concrete one, we build domain specific tools to do this more efficiently. At the beginning of the tool development, we lift the classes from the abstract domain model into a meta-model. This meta-model can be seen as a data-model to store and interchange domain specific models [Sei03]. It is used as data-model for a model-editor and a code-generator. The model editor is used to create and edit domain specific models that are transformed into a concrete domain model by the code generator. Both these tools could be developed following traditional software development methods. However, there exist frameworks like GMF and oaw [Ecl09] that facilitate this development significantly.

This being said, we have two separate places where we can model the different aspects of a software system: the abstract domain model and the meta-model. For every aspect

we may decide to either model it in the abstract domain model with a manual implementation possibly completed by an extension of the code generator, or to model it in the meta-model with its implementation becoming an integral part of the code generator. The most significant difference between domain modeling and meta-modeling in our approach is changeability. An aspect modeled in the meta-model will usually be transformed into static code. Thus, a change here will have us to deploy a new version of the software. Modeling the same aspect in the abstract domain model usually allows us to make changes at runtime. To get there, modeling of behavior at this level will force us to implement complex interpreters.

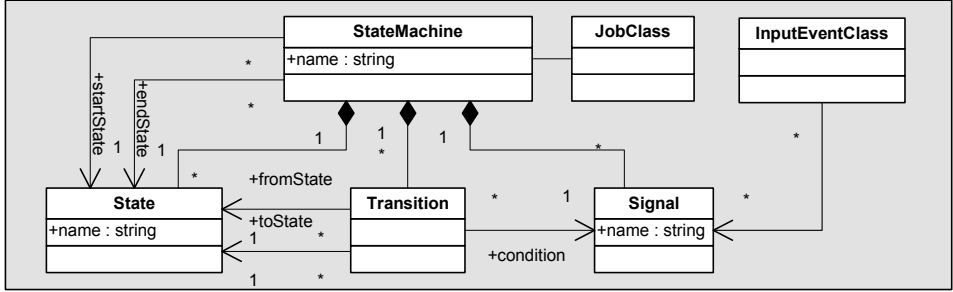


Figure 8: Extension of the meta-model to allow the modeling of state machines for jobs.

As an example for an extension of the meta-model, we present the modeling of the state of a job. A job can have different states at different points in time. Input events may cause jobs to change their state. The possible states of a job and the transition between them with the respective conditions form a state machine. In a production system, we may not change the state machine for a type of job as this will falsify the history of past jobs. Change can only be accomplished by adding a new type of job with a new state machine. Hence we model the state machines in the meta-model and generate static code for them. Figure 8 contains the corresponding meta-model extension and Figure 9 contains the domain specific model with the state machine for ProductionJobs.

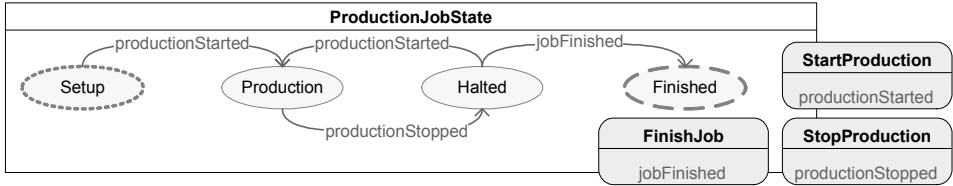


Figure 9: Model of the different states for ProductionJobs.

On the other hand, the history of a job as a series of slots is modeled in the domain model (Figure 10). Each Slot stores the state of its job in the period of time before its startEvent and its endEvent. Besides this data structure for slots, we need a structure to make some statistics over the usage of time for a given job. The abstract class BaseTimeDistribution allows to sum up the total time of the slots that are given to it. In our example we are interested in how long a job spend in its different states. Therefore we generate the ProductionJobsDistribution for ProductionJobs that contains distribu-

tions for each one of the possible job states. When adding slots to this ProductionJobsDistribution, the slots are recursively added to the distributions corresponding to the job's state stored in the slot. As all information needed to generate these TimeDistributions is already contained in our model, there is no need to extend the meta-model.

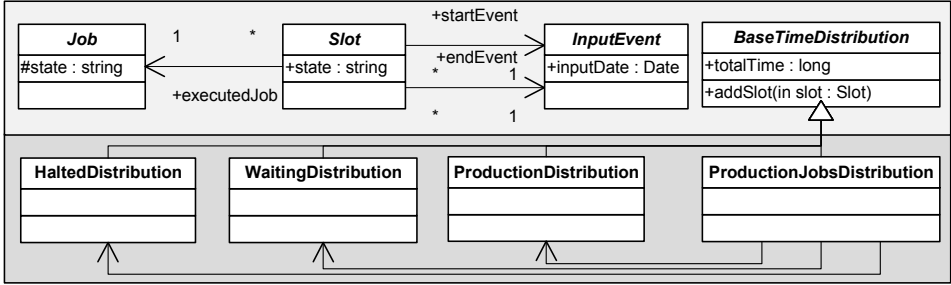


Figure 10: Extension of the framework-model to build statistics on the time a job spent in different states.

If we had modeled the VariableUpdateRelation or the state machine in the domain model, we would have needed to implement interpreters to execute the stored update relations or state transitions.

6 Experimental Results

We implemented the whole MDE framework for MES, including meta-model, model-editor and code-generator using Eclipse Modeling Framework (EMF), Graphical Modeling Framework (GMF) and openarchitectureware (oaw) [Ecl09]. As target platform we chose Java 5 with Enterprise Java Beans (EJB) 3.0. The structure of the code generator follows the principles of the Model Driven Architecture (MDA) [OMG03], i.e. it consists of several modules responsible for different architectural elements (CIM to PIM) and each module generates the needed Java code (PIM to PSM to code).

For the implementation of the GUI we used QT-Jambi and for the generation of reports we used Crystal Reports. Both tools fit well in our model driven approach with their good WYSIWYG designers.

The generated system has been tested in a real world environment using the JBoss application server at an injection molding company. During the testing phase, several bugs have been identified that could easily be fixed in the framework and code-generator. Some feature requests arose that could be implemented quickly by small additions to the meta-model and code-generator. The most difficult parts have been the GUI-enhancements to make the new features visible and editable.

The implementation of the backend of our Manufacturing Execution System (Figure 2) contains 1541 lines of code in 45 classes, the complete MES meta-model contains 22 meta-classes with 151 model elements (=classes, attributes, and associations) and the code-generator templates contain 4356 lines of code. The complete model of the manu-

facturing execution system for the injection molding company is depicted in Figure 11. It contains 175 model elements. From these we generate 90 classes with a total of 7525 lines of code that are recreated every time the generator is run. Another 37 subclasses with 594 lines of code are created once at the first generator run. These may be manually modified to implement some custom functions. In our system, we added a total of 128 lines of code to 21 of these classes. This gives us a fix cost of 151 model elements plus 5897 lines of code and a variable cost of 175 model elements plus 128 lines of code for this one system.

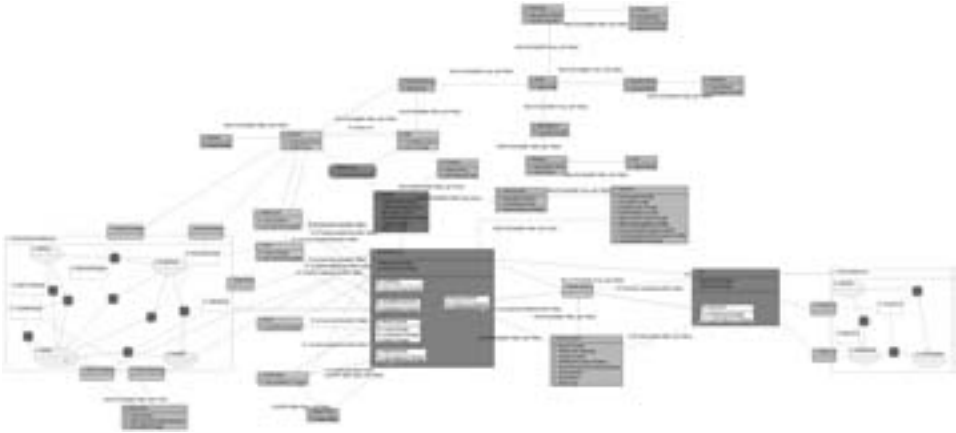


Figure 11: Complete Model of the implemented MES-System.

The overall system architecture would not be different if we had not chosen to use code generation. Thus, we need to compare the effort to write the 8119 generated lines of code with the effort to write the 4356 lines of code in the code generator and the 326 model elements. Around half the lines of code in the code generator are final code templates. This means, that each code template is reproduced around four times in average. In consequence, only around a quarter of the generated code has to be tested to be sure that “all” code works correctly whereas the manual code has to be tested completely. Additionally a change in one implementation pattern due to an error affects around four code blocks that need to be changed independently but consistently in the manual code. The renaming of a model element in compliance with all naming conventions might propagate to even more different places. Even if manually writing these lines seems attractive, testing them appropriately and keeping them consistent over time will be very hard.

Besides the slight numerical gain in productivity in the development of this first system, the MDE approach will speed up the development of further systems due to the small variable cost per system. Additionally there is a huge increase in maintainability due to the automatically assured consistence between all code parts involved in the handling of the same objects. The possibility to systematically add new functionality either in the manually implemented framework (domain modeling) or in the code generator (meta-modeling) makes the systems easily extensible. Finally, the validation of the code gen-

erator templates by testing a “minimal” system guarantees the quality of a complete system.

7 Conclusion

We presented an approach to model driven engineering, where the central meta-model is defined starting from a traditional abstract object-oriented domain model. Using the differences between modeling at the abstract domain level and modeling at the meta-level, we may integrate both levels and choose for each function of a software system the optimal way of modeling and implementation.

The models conforming to a meta-model defined following our method can be used to generate a concrete specialization of the underlying abstract domain model where all elements have an assured quality and are guaranteed to be consistent with one another. Thus, our method achieves the most important goals of model driven software development such as higher productivity, easier extensibility and better quality.

As meta-modeling is the most difficult and discouraging step in model driven engineering, the concrete explanations in our method about how to start the definition of a meta-model, what to put into the meta-model and what to keep in the abstract domain model, make the access to MDE easier for experienced object-oriented developers.

References

- [AK03] Atkinson, C., Kühne, T., 2003. Model-driven development: a metamodeling foundation. In *IEEE Software*. Vol. 20. No. 5, pp 36-41. September-October 2003.
- [AC06] Antkiewicz, M., Czarnecki, K., 2006. Framework-Specific Modeling Languages with Round-Trip Engineering. In *LNCS Vol. 4199*. Springer. Berlin / Heidelberg. 2006.
- [Ecl09] Eclipse Modeling Project EMF, GMF, oaw: <http://www.eclipse.org/modeling/http://www.openarchitectureware.org/> (accessed 18th may 2009)
- [Ev04] Evans, E., 2004. *Domain-Driven Design Tackling Complexity in the Heart of Software*. Boston. Addison-Wesley.
- [Fo03] Fowler, M, 2003. *Patterns of Enterprise Application Architecture*. Boston. Addison Wesley.
- [He06] Hesse, W., 2006. More matters on (meta-)modeling: remarks on Thomas Kühne’s “matters”. In *Journal on Software and Systems Modeling*, Vol. 5, No. 4. pp. 387-394. December 2006.
- [Ke02] Kent, S., 2002. Model Driven Engineering. In *Proceedings of the Third International Conference on Integrated Formal Methods*. LNCS Vol. 2335, pp 286 – 298. 2002.
- [KI07] Kleppe, A., 2007. A Language Description is More than a Metamodel. In *4th Int. Workshop on Software Language Engineering*. Nashville. USA. October 2007.
- [KK03] Koch, N., Kraus, A., 2003. Towards a Common Meta-model for the Development of Web Applications. In *LNCS Vol. 2722*. Springer. Berlin. 2003
- [Kü06] Kühne, T., 2006. Matters of (Meta-)Modeling. In *Journal on Software and Systems Modeling*, Vol. 5, No. 4. pp. 369-385. December 2006.
- [LKT04] Luoma, J., Kelly, S., Tolvanen, J-P., 2004. Defining Domain-Specific Modeling Languages: Collected Experiences. In *Proc. of the 4th OOPSLA Workshop on Domain-*

Specific Modeling (DSM'04), Technical Reports, TR-33, University of Jyväskylä, Finland 2004

- [MHS05] Mernik, M., Heering, J., Sloane, A. M., 2005. When and How to Develop Domain-Specific Languages. In ACM Computing Surveys (CSUR). Volume 37. Issue 4. pp. 316-344. December 2005.
- [MFM08] Mohagheghi, P., Fernandez, M. A., Martell, J. A., Fritzsche, M., Giliani, W., 2008. MDE Adoption in Industry: Challenges and Success Criteria. In 1st Int. Workshop on Challenges in Model Driven Software Engineering. September 28th. Toulouse. France.
- [OMG03] Object Management Group (OMG), 2003. MDA Guide Version 1.0.1, OMG Document/03-06.-01. 2003.
- [PD08] Posse, E., Dingel, J., 2008: A Foundation for MDE. In 1st Int. Workshop on Challenges in Model Driven Software Engineering. September 28th. Toulouse. France.
- [RJ96] Roberts, D., Johnson, R., 1996. Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks. In 3rd Conference on Pattern Languages and Programming. Addison-Wesley. 1996.
- [RR08] Rutle, A., Rossini, A., 2008. A Tentative Analysis of the Factors Affecting the Industrial Adoption of MDE. In 1st Int. Workshop on Challenges in Model Driven Software Engineering. September 28th. Toulouse. France.
- [Sa07] Santos, A. L., 2007. Automatic Support for Model-Driven Specialization of Object-Oriented Frameworks. In 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion. pp. 923-924. Montreal. Quebec. Canada. 2007
- [Sei03] Seidewitz, E., 2003. What Models Mean. In IEEE Software. Vol. 20. No. 5, pp 26 – 32. September 2003.
- [Sel08] Selic, B., 2008. Personal Reflections on Automation, Programming Culture, and Model-based Software Engineering. In Automated Software Engineering. Vol. 15. Issue 3-4. pp 379 – 391. December 2008.
- [TG08] Thörn, C., Gustafsson, T., 2008. Uptake of Modeling Practices in SMEs Initial Results from an Industrial Survey. In 2008 Int. Workshop on Models in software Engineering. Leipzig. Germany.
- [DKV00] van Deursen, A., Klingt, P., Visser, J., 2000. Domain Specific Languages. In ACM SIGPLAN Notices. Volume 35. Issue 6. pp. 26-36. June 2000.
- [WS07] Weisemöller, I., Schürr, A., 2007. A Comparison of Standard Compliant Ways to Define Domain Specific Languages. In LNCS Vol. 5002, pp. 47-58. Springer. Berlin / Heidelberg 2008.

Integrating System Dynamics with Object-Role Modeling and Petri Nets

P. (Fiona) Tulinayo¹, S.J.B.A (Stijn) Hoppenbrouwers¹,
Patrick van Bommel¹, H.A. Erik Proper^{1,2}

¹Institute of Computing and Information Sciences, Radboud University Nijmegen
Heyendaalseweg 135, 6525 AJ Nijmegen, The Netherlands.

²Capgemini Nederland B.V.,

Papendorpseweg 100, 3528 BJ Utrecht, The Netherlands.

F.Tulinayo@science.ru.nl, stijnh@cs.ru.nl, pvb@cs.ru.nl, e.proper@acm.org

Abstract: The art of System Dynamics (SD) modeling lies in discovering and representing the feedback processes and other elements that determine the dynamics of a system. However, SD shows a lack of means for discovering and expressing precise, language-based concepts in domains. Therefore, we choose to use Object-Role Modeling (ORM) to add high quality formal conceptualization to SD modeling and Petri Nets (PNs) to bridge the gap between static ORM and Dynamic, flow-like aspects of SD. To achieve the integration of these methods, we take a step by step approach where we first identify the conceptual link between SD and ORM, then the key concepts used in these methods, which we later use to derive mappings, transition statements and elements. This helps us to better understand the underlying concepts, the connections between the model structure, and behavior in a sequential manner.

1 Introduction

Integration of methods is an approach that can be applied to complex software development, it aims for combined use of different (generic) methods for highly specific purposes [Att00]. Paige [Pai97] defines method integration as an involvement in defining relationships between different methods so that they may be productively used together to solve problems. He further gives more definitions inline with method integration in [Pai99]. In this paper we integrate System Dynamics [For61] with object-role modeling (ORM) [Hal98] and Petri nets [Mur89] i.e extract different views of the same model, compare and relate them among themselves and then combine them. Our main interest is to improve SD modeling by deploying methods and techniques from system development (ORM and Petri Nets).

To model the dynamics of key variables within a process, we use SD a method that has been in existence since 1961, developed by Jay Forrester to handle socio-economic problems with a focus on the structure and behavior of systems composed of interacting feedback loops. A review and history is given in [For61]. As a method, it has its focus on the structure and behavior of systems composed of interacting feedback loops, it also provides

a high level view of the system emphasizing the interactions between its constituent parts, as well as the impact of time on its dynamic behavior [HGM01]. The dynamics arise from the interaction of two types of feedback loops, positive and negative. Positive loops tend to reinforce or amplify whatever is happening in the system while negative loops counteract and oppose change. These loops all describe processes that tend to be self limiting, create balance and equilibrium [PL99].

There have been earlier comparative studies, concerning methods potentially complementary to SD, for example SD and Discrete-event system [BH00], [BF04]; and SD and Petri nets [Dug06]. In these comparisons the main differences between SD and these methods have been highlighted but they do not state the static conceptualization of system/process which this research looks at.

For strong verbalization, conceptualization and a fully formal link to predicate logic we use ORM which is comparable to Entity relationship (ER) diagrams in use [Che76]. It is, however, a *fact-oriented* approach for modeling information at a conceptual level [HW03]. It was initially developed in the early 70's [Hal98] and has a graphical constraint notation that is claimed to be far more expressive for data modeling purposes than, for example, Unified Modeling Language (UML) class diagrams or industrial Entity Relationships (ER) diagrams [HW03]. Halpin and Wagner do state that *'although ORM supports modeling of business terms facts, and many static integrity constraints and derivation rules, it cannot model the reactive behavior of systems which can be described using dynamic integrity constraints'*. Clearly we use SD to capture the reactive behavior of a system. [Sha05] further states that; *"....there is a strong case for starting to apply systems dynamics methods more openly in the BPM and MIS research fields, as I feel the tools and techniques available are vastly under-rated in terms of their applicability and capability to provide novel representations of real-world situations....."*. These complementary statements are the basis for our overall research direction: integration of SD with Conceptual and Process Modeling.

PNs which are abstract, formal models of information flow that were originally developed to model causal relationships between asynchronous components of a computer system [Dug06] are also used. They have been around for a considerable time and are widely used for modeling workflow systems. Various scholars e.g. [Pet81]; [Mur89] give detailed studies and their background ¹.

To give an insight of the overall view of the study, we came up with Fig.1 where we sketch how we are to integrate the three methods (SD, ORM and PNs). In this illustration, two types of mappings are shown: mappings between viewpoints are what we refer to as inter-viewpoint mappings, and the mappings between specific viewpoints and integrated meta model are refereed to as viewpoint meta model mappings. We use ORM, which is a state and event reporting, fact oriented modeling technique [HW03], as a graphical representation for conceptual structure. PNs are used to model a discrete flow (in a sense that they contain sequential quantities) , and SD to model a quantitative flow (items or quantifications that flow with in the system). With these three models integrated we hope

¹Note: all methods used in this paper are in their simplest form, this is because when dealing with complex or different methods the modeler needs to understand the underlying concepts, connections and behavior in their simplest form before moving into details. As [Ric96] states *"understanding connections between model structure and behavior comes from a sequential modeling process that is from simpler formulations to complex structures"*

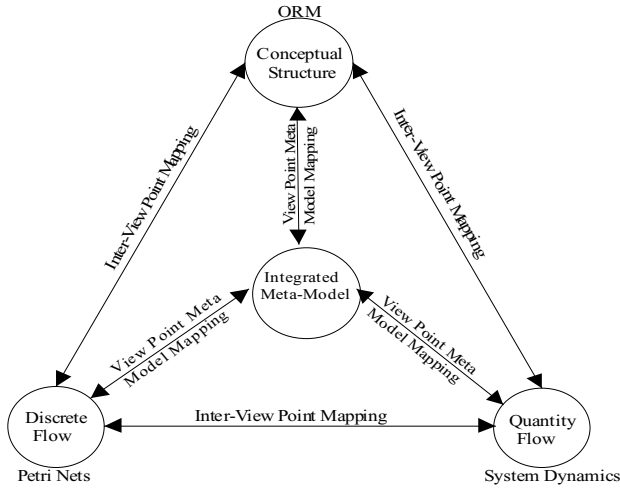


Figure 1: Abstract View of the Integration

to produce better understood models all-round, though we emphasize the use of ORM and PNs in the service of SD modeling.

The structure of this paper is as follows. In section 2 we explain the basic concepts of the three methods, then explore the conceptual links between SD and ORM; and finally SD, ORM, and PNs. We also present a step by step schema based approach for transforming an ORM domain model into an SD and PN model. In section 2.3 we use the same example to come up with a causal Loop Diagram (CLD) and Stock and Flow Diagram (SFD). In section 3 we identify the commonly used variables in the three methods, state their importance, and how they are used in the methods. We present this in two tables. In the first table we map two of the methods indicating their transitions. In the second table we add a third method (PNs), against each concept we put a transition statement to explain the similarities among these concepts. Thus this paper presents an exercise in usefully linking three existing and rather different methods in enterprize modeling.

2 Conceptual Link between SD and ORM

Before we conceptually link these methods, Lets start by explaining the basic underlying concepts of the three methods, on which we base our studies or assumptions.

The key variables we use in this paper are; System Dynamics under (A), Object Role modeling under (B), and PNs under (C) depicted in Fig.2. In SD we use *Stocks* (Stock A and Stock B), *Information links*, *Exogenous variables*, and *Flow rates* (Inflow into stock A and Outflow from Stock A into stock B). Stocks can be considered reservoirs containing quantities describing the state of the system. Flows (inflow to and outflow from the various levels) can be imagined as pipelines with a valve that controls the rate of accumulation to and from the stocks. The Exogenous variables contain information in the form of equations

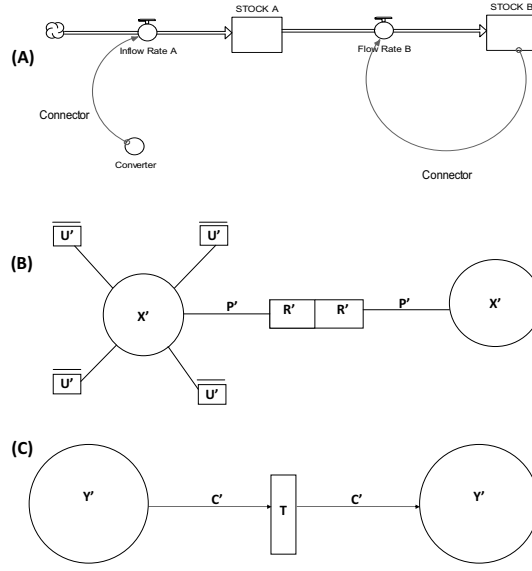


Figure 2: SD,ORM and Petri Nets Concepts

or values that can be applied to stocks, flows, and other Exogenous variables in the model [LU07].

Information links and Exogenous variables measure the quantities in levels and, through various calculations, control the rates. They appear as lines with arrows (Information links) and as circles (Exogenous variables).

In the conventions for a stock and flow diagram: (a) Information links can feed information into or out of flows and Exogenous variables but only extract information out of the stock. (b) Stocks are influenced by flows (in and out) and can influence flows or Exogenous variables but cannot be influenced by other stocks and Exogenous variables. (c) The flows can be influenced by stocks and Exogenous variables but cannot influence Exogenous variables or other flows, and Exogenous variables can influence flows or other Exogenous variables.

In ORM we use *object types* to denote entities (at type level) and *fact types* as predicate-like relationships between object types (X'). Object types are a collections of objects with similar properties, in the set-theoretical sense. Fact types (R' , U') represent associations between object types, consisting of a number of roles denoting the way object types participate in that fact type. We can have, for example, *unary fact types* (U') and *binary fact types* (R').

For PNs, *Places* are denoted as (Y') which are inactive concepts analogous to in-boxes in an office workflow system. They are depicted as circles, each PN has one start place and one end place, but a number of intermediate places. *Transitions* (T) are active and represent tasks to be performed. They are depicted as rectangles in Fig.2. *Arcs* (C') are shown as connecting lines. An inward arc goes from a Place to a Transition and an outward

arc goes from a Transition to a Place. Tokens exist within Places and represent the current state of a process.

2.1 Using ORM as a Foundation for SD Models and Modeling Processes

The first step in our approach consists of three sub-steps in which an ORM diagram is augmented and replaced by process concepts that lean towards SD-like conceptualization. The second step consists of two sub-steps that concern the construction of actual CLD and SFD. We use as a working example the procedures a paper might go through en route from writing to publication (*Note that this example does not illustrate the full power of SD*). The procedures are stated as:

1. A person (author) writes an intent of submission. This can be in the form of an abstract.
2. Then the content (text of the paper) is submitted, whereby the paper becomes a submitted paper.
3. Each submitted paper receives a classification.
4. Each submitted paper is reviewed.
5. Some submitted papers are accepted and some are rejected.
6. For each accepted paper new content is submitted, which makes the paper a published paper that is added to the publications.

These statements can be represented in an ORM diagram as indicated in Fig.3:

Step 1.(a) We start with Fig.3 which is an ORM diagram of events (reported as elemen-

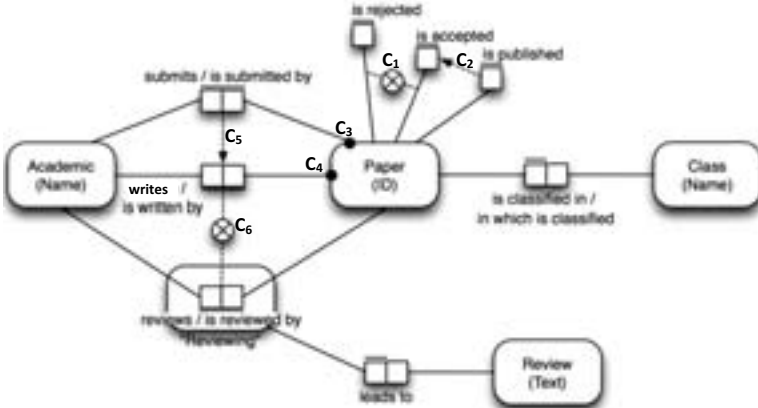


Figure 3: Paper flow concepts in ORM

tary facts) that may be observed in a domain (in this case, the reviewing domain). This approach is inline with the PSM^2 [vBFvdW97] approach. The symbol and Constraint used in fig.3 can be Verbalized as follows:

- C_1 (Exclusive): *each* paper plays *exactly one of the roles* is rejected or is accepted.
- C_2 : *each* paper published is *an* accepted paper.
- C_3 (mandatory): *each* paper is submitted by *at least one* academic.
- C_4 (mandatory): *each* paper is written by *at least one* academic.
- C_5 : *each* Academic *who* submits *a* paper *also* writes *that* paper.
- C_6 (Exclusive): *each* academic plays *exactly one of the roles* reviews or writes.

2.2 ORM integrated with SD and petri Nets

Step 1.(b) We add temporal dependencies between the roles associated to the paper. This leads to the flow depicted in Fig.4. The left hand side depicts the full diagram based on the facts types (event types) in the original ORM diagram. The diamond shape is the BPM [Whi04] symbol for an XOR split. Our focus is on the flow statics of submissions, reviews, acceptance, rejection and publication. This is a section of the roles connected to the object types in Fig.3. This leads to the abstracted view depicted on the right hand side.

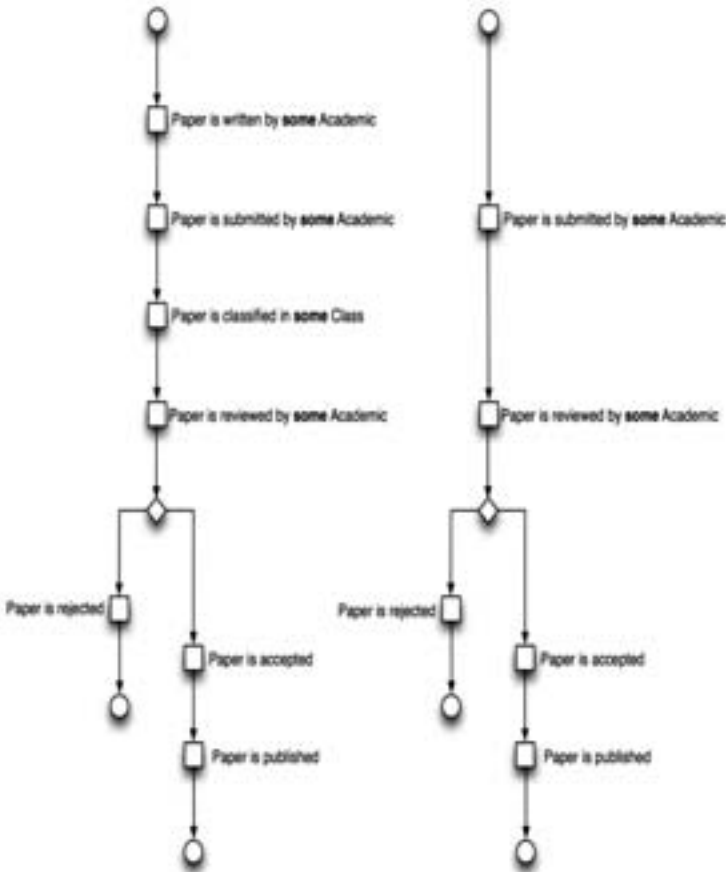


Figure 4: Paper Flow

Step 1.(c)we now make explicit the relations between, in particular, the ORM model and stock-flow model.

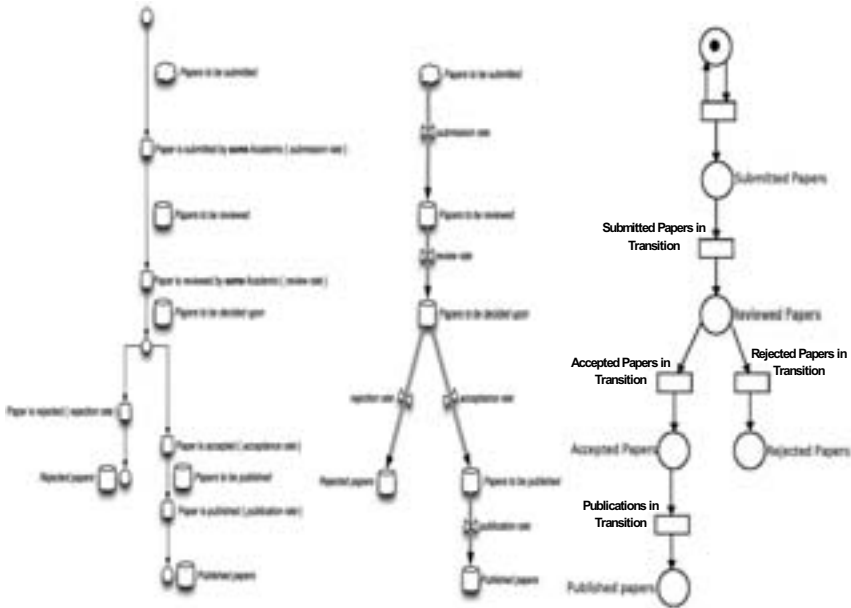


Figure 5: An integration of ORM, SD and Petri nets

In Fig.5 the left hand side shows Fig.4 (right) with some extra information and the right hand side is the Petri Net model. The extra information pertains to the flow-based interpretation. We now see stores of papers that are ready to flow from one state to another. Each time a paper "flows", this is an event (the original events as related to ORM the diagram, Fig.3). So:

- A paper is **reviewed**
- A paper is **decided upon**

Associated to the event-types, we can now also add a rate. Leading to:

- Review rate
- Acceptance rate

The model in the center depicts the SD diagram. This is the prelude to the complete SD Stock and Flow diagram as depicted in Fig.8. Note that the SD model contains five stocks, which are caused by our chosen focus on submissions, reviews, acceptance, rejection and publication explained under Step.1(b).

The PN model on the right (Fig.5) shows that every time a token (paper) is fired (submitted papers) another token is replaced. It goes through a transition (submitted papers in transition) to get to the next place (Reviewed papers) [*This holds for all places and transitions*]

in this example]. Here we start to see that the transitions (PN) and Flows (SD) carryout similar jobs of transferring token (PN)/Quantities (SD) from one place (PN)/Stock (SD) to another place (PN)/Stock (SD). More of these findings are given in Tables 1 and 2.

2.3 Causal Loop Diagram, and Stock and Flow Diagram

Step 2.(a) We now embark on identifying the key variables for the SD model. In SD, two diagrams are most commonly used: causal loop diagrams (CLD) and stock-flow diagrams. A CLD (in our case, created using Vensim simulation software) helps us show the main feedback loops (feedback is defined as the transmission and return of information [RP81]) in a process. Below is a clear description of a CLD or feedback loop.

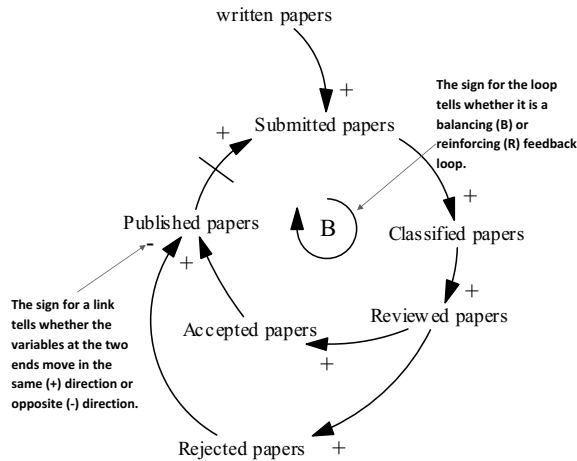


Figure 6: A causal Loop Diagram for paper flow

Figure. 6 is an annotated causal loop diagram for a simple process a paper might go through from writing to publication. This diagram includes elements and arrows (which are called *causal links*) but also includes signs (either + or -) on each link. These signs have the following meanings:

1. A causal link from one element A to another element B is *positive* (that is, +) if either (a) A adds to B or (b) a change in A produces a change in B in the *same* direction.
2. A causal link from one element A to another element B is *negative* (that is, -) if either (a) A subtracts from B or (b) a change in A produces a change in B in the *opposite* direction.

Starting from the element “*Submitted papers*” at the top of the diagram. If the Submitted papers increase then the “*Classified Papers*” also increase. Therefore, the sign on the link from “Submitted papers” to “Classified Papers” is positive. Next, if the “Classified Papers” increase, then the “Reviewed papers” increase. Therefore, the sign on the link

between these two elements is positive. Similarly, if the “*Reviewed papers*” increase, then the “*Accepted Papers*” increase. Hence, the sign on the link between these two elements is also positive. The Increase in “*Accepted Papers*” causes an increase in “*Published papers*”. The next element along the chain of causal influences is the “*Delay*,” this is because when there is an increase in “*Published Papers*”, there is a lag (delay) before the actual increase in “*submitted papers*” is noticed (evident) leading to a balanced loop (B). On the other hand if the “*Reviewed papers*” increase, then the “*Rejected Papers*” increase. Therefore, the sign on the link between these two elements is also positive. The Increase in “*Rejected Papers*” causes a decrease in “*Published papers*”. Therefore, the sign on the link from “*Rejected Papers*” to “*Published Papers*” is negative.

In addition to the signs on each link, a complete loop also is given a sign. The sign for a particular loop is determined by counting the number of minus (-) signs on all the links that make up the loop. Specifically,

1. A feedback loop is called *positive (B)*, indicated by a + sign in parentheses, if it contains an even number of negative causal links.
2. A feedback loop is called *negative (R)*, indicated by a -sign in parentheses, if it contains an odd number of negative causal links.

Thus, the sign of a loop is the algebraic product of the signs of its links. Often a small looping arrow is drawn around the feedback loop sign to more clearly indicate that the sign refers to the loop.

Step2.(b) As with a causal loop diagram, the SFD shows relationships among variables which have the potential to change over time. The SFDs have four different concepts [*Stocks, Flow-Rates, Information links (Connectors)* and *Exogenous variables (Converters)*].

The SFD is constructed here using the Powersim application. This is a simulation tool based on the SD methodology. The SFD was used to show flow dependencies and how quantities are distributed within the system. Stocks hold quantities that are subject to accumulation through inflows, or to reduction through outflows. We have the stocks as submitted papers, papers to be decided upon, rejected papers, accepted papers, and published papers. These stocks have inflows and outflows that are regulated by means of valves. The valves determine the rate at which an inflow or outflow of material applies to the stock (box). In this model there are different factors that affect the flows, and these are either positive or negative. These effects can be indicated as constants linked to the flows or stocks with a Information links. During the development of the stock and flow model a number of experimental simulations are normally run to show the different behaviors of the system studied. In our case, such simulations were also carried out, on selected simulation parameters. While doing so, the model can be paused, and each of the stocks continues to hold its quantity for observation. If the value of a particular stock is not important to the problem at hand, then it is shown as a cloud, to indicate that it is outside the boundary of the model. This procedure is also known as sensitive analysis [MR08]. From Fig.7 each rate is clearly defined as follows;

Let X and Y be the input and output of some flow $X \Rightarrow Y$:

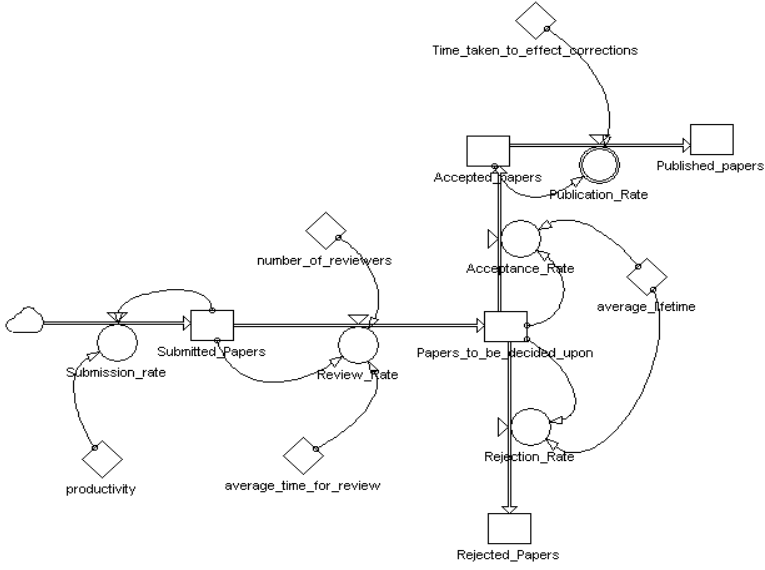


Figure 7: A stock and flow diagram for the paper Procedure

Then the rate for this flow at time t is defined as

$$\text{Rate}(X \Rightarrow Y) \frac{\text{TotalPop}_t(Y)}{\text{TotalPop}_t(X)}$$

Where $\text{Totalpop}_t(X)$ is the set of all instances ever of stock.

For cases where we have more outflow rates from the same stock and more inflow rates from different stocks we would have for each stock X :

$$\text{TotalPop}_t(X) \text{Pop}_t(X) \cup \bigcup_{Y: X \Rightarrow Y} \text{AddedFlow}_t(X, Y)$$

$$\text{AddedFlow}_t(X, Y) \text{TotalPop}_t(Y) - \bigcup_{X': X' \Rightarrow Y \wedge X' \neq X} \text{TotalPop}_t(X')$$

$$\text{TotalPop}_t(X) \text{Pop}_t(X) \cup \bigcup_{Y: X \Rightarrow Y} \left(\text{TotalPop}_t(Y) - \bigcup_{X': X' \Rightarrow Y \wedge X' \neq X} \text{TotalPop}_t(X') \right)$$

Note that in general this will lead to a recursive system of equation. As all instances have assigned unique locations at each moment, this recursive system of equations will have a unique location.

3 Mappings between Methods

In this we use the key variables identified in figure.2 section.2 to make explicit the relationships among these methods. This helps us in mapping the different concepts used in the methods. Apart from identifying the connections or justifiable similarities among these methods, we note their transitional statements and elements. The relationships among these methods are derived from the way concepts interact amongst themselves, or the roles they play in the process of modeling systems. Table.1 is used as starting point for the integration or combination of these techniques. We note one aspect of the methods being integrated that's SD modeling as explained earlier has dynamic properties while ORM has static properties. Static properties refer to the possible states of the system under study while dynamic properties refer to the possible transitions between the states [HEG93]. This raises an interesting question: how can we integrate a static method with a dynamic method? Our static method is ORM², which describes the objects that make up the states of the system as well as the integrity constraints on the states. The dynamic side of things is represented by SD [For61] and Petri nets [Pet81]. In Petri Nets, terms like Place, Transition (Link), Token and Arc are used, while in SD; Stock, Flow, information links, and exogenous variables are used. These are two completely different worlds but they both model dynamic systems.

Table 1: SD, ORM plus their transitional statements

| System Dynamics | ORM | Transitional Statement |
|--------------------------------|------------------|---|
| Stock | Unary fact types | They all contain "things" or act as containers. |
| Quantity | Objects | These can be looked at as the contents within the system. |
| Flows (Inflow and Outflow) | Object types | They all connect different stocks (SD) and Fact types (ORM). |
| Information links (connectors) | Fact types | They are both active and have movements involved that cause a change to the recipient or destination. |

When mapping the different concepts used, we found connections among these concepts. *Stocks* in SD are similar (though not identical) to *unary fact types* in ORM because they both contain "*things*"(quantities for SD and Objects for ORM). *Quantity* in SD are similar to counting *Objects* in ORM. This is because we look at them as quantities that flow within the system or process. We use the term "quantity" in SD to represent the items or quantifications that flow with in the system. Next we link *flows* (inflows and out flows) in SD to *Object types*. This is because they connect different stocks (SD) and Unary fact

²Yet also Petri Nets which has been developed and used mainly in modeling dynamic systems and processes; they are also suitable for modeling static properties although the diagrams produced are big and complex [HEG93].

types (ORM) respectively. Finally *Information links* are linked to *fact types* because they are both active and have activities involved that cause a change to the recipient/destination. Fig.2 (A) System Dynamics, (B) Object-Role Modeling and (C) Petri nets depict the discussed variables in this paragraph.

Table 2: SD with ORM and Petri nets

| System Dynamics | ORM | Petri nets | Transitional Statement | Elements |
|--------------------------------|------------------|-------------|--|--------------------------|
| Stock | Unary fact types | Places | They all contain “things” (quantity (SD), Objects (ORM) and Tokens (PNs)) | Containers |
| Quantity | Objects | Tokens | These can be looked at as the things that flow with in the system | Contents |
| Flows (In-flow and Outflow) | Object types | Transitions | They all connect and transfer items from one state to another state (<i>in a sequential manner</i>): Stocks (SD), Unary fact types, (ORM) and Places (PNs) | Homogeneous connectors |
| Information links (connectors) | Fact types | Arcs | They are all active and involve activities that cause a change to the recipient or destination | Heterogeneous connectors |

In Table.2 *Stocks* in SD are similar (though not identical) to *unary fact types* in ORM and *Places* in PNs because they all act as containers of quantities (SD), Objects (ORM) and tokens (Petri Nets). We refer to their elements as *containers* because of their purpose which is holding items. *Quantity* in SD are similar to *counting Objects* in ORM and *Tokens* in PNs because they are the “Things/contents” that flow within the system or process and are held in stocks/places/unary fact types. we refer to their elements as *contents*. *Flows* (inflows and out flows) in SD are linked to *Object types* in ORM and *Transitions* in Petri nets because they all connect different stocks (SD), Unary fact types (ORM) and Places (Petri Nets), and transfer objects (ORM)/tokens (PN)/ Quantity (SD). We refer to their element as Homogeneous connectors because they all connect and transfer similar concepts. Finally we link *information links* (SD), fact types (ORM) and Arcs (Petri Nets) because they are all active and have activities involved that cause a change to the recipient. Although, transfers are made through them, they do not carry similar items that's why we refer to their elements as Heterogeneous connectors.

4 Conclusion and Further Research

In this paper we have identified the key features in three modeling methods, and mapped them to show their relationships, transitions and elements. We have shown the extent to which the features of ORM static models can be transformed (with added information)

into Petri Net and SD models. The ORM methodology equips the modeler with strong conceptualization of the domain which is key in developing any model. By combining SD concepts with ORM and Petri Nets style modeling, we manage to better capture the static part of the model, and to link it with the dynamic aspect.

This research is part of larger project aiming at improving SD modeling by deploying methods and techniques from system development. An expectation is that the models produced this way are better understood with less errors than is currently the case. This will have to be empirically confirmed. With higher quality SD models in place, decision makers and stakeholders should be able to make better decisions concerning their enterprise and its processes. We will apply the approach presented in context of various case domains. We will further develop and refine the method (its models as well as the stepwise process): By devoting more attention to formalizing its syntax and semantics, but also to operationalization of the modeling procedures. In addition, we intend to use the techniques suggested in this paper in collaborative settings (Group Model Building, [RH08]), which is a sub discipline within the field of SD. Finally, we intend to explore further links between SD and process modeling (already initiated by the Petri Net involvement).

References

- [Att00] C. Attiogb'e. Formal Methods Integration for Software Development: Some Locks and Outlines. In *Technical Report: 00.8*. IRIN, University of Nantes, 2000.
- [BF04] A. Borshchev and A. Filippov. From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools. *22nd International Conference of the System Dynamics Society*, 2004.
- [BH00] S.C. Brailsford and N.A. Hilton. A Comparison of Discrete Event Simulation and System Dynamics for Modelling Healthcare Systems. *Proceedings of ORAHS, Glasgow Caledonian University*, pages 18 – 39, 2000.
- [Che76] P.P. Chen. The entity-Relationship model-Towards a unified view data. *ACM Transactions of database systems*, 1(1):9–36, March 1976.
- [Dug06] J. Duggan. A Comparison of Petri Net and System Dynamics Approaches for Modelling Dynamic Feedback Systems. *24th International Conference of the Systems Dynamics Society, Nijmegen, The Netherlands*, July 2006.
- [For61] J.W. Forrester. *Industrial Dynamics*. Productivity Press; Student Edition edition, edition, 1961.
- [Hal98] T. Halpin. 'Object-Role Modeling (ORM/NIAM)', *Handbook on Architectures of Information Systems*. Springer Berlin Heidelberg, 1998.
- [HEG93] C. A. Heuser, Meira Peres. E, and Richter. G. Towards a complete conceptual model: Petri nets and entity-relationship diagrams. *Information Systems*, 18(5):275 – 298, July 1993.
- [HGM01] J.C. Hustache, M. Gibellini, and P.L. Matos. A System Dynamics Tool for Economic Performance Assessment in Air Traffic Management. *4th USA/Europe Air Traffic Management R and D Seminar*, pages 3 – 7, December 2001.

- [HW03] T Halpin and G. Wagner. Modeling reactive Behavior in ORM. *LNCS* , Springer-Verlag Berlin Heidelberg, 2813:567–569, October 2003.
- [LU07] J.D Leaver and C. P. ; Unsworth. System dynamics modeling of spring behavior in the Orakeikorako geothermal field. *Elsevier Science, Oxford, ROYAUME-UNI*, 36(2):101–114, April 2007.
- [MR08] B. Mutschler and M. Reichert. On Modeling and Analyzing Cost Factors in Information Systems Engineering. In *Advanced Information Systems Engineering*, pages 510–524. Springer Berlin / Heidelberg, June 2008.
- [Mur89] T. Murata. Petri Nets: Properties, analysis and applications. *Proc. of the IEEE*, 77:541–580, April 1989.
- [Pai97] R. Paige. A meta-method for formal method integration . *Proc. Formal Methods Europe, LNCS*, 1313:473–494, September 1997.
- [Pai99] R. Paige. When are methods complementary? . *ISM*, 41(3):157–162, February 1999.
- [Pet81] J.L. Peterson. *Petri Net Theory and the Modeling of System*. Prentice-Hall PTR, Upper Saddle River, NJ, USA, edition, 1981.
- [PL99] D. Pfahl and K. Lebsanft. Integration of system dynamics modeling with descriptive process modeling and goal-oriented measurement. *Journal of Systems and software*, 41:135–150, April 1999.
- [RH08] E. Rouwette and S.J.B.A. Hoppenbrouwers. Collaborative systems modeling and group model building: a useful combination? *26th International Conference of the System Dynamics Society*, December 2008.
- [Ric96] G. Richardson. Problems for the future of system dynamics. *System Dynamics Review*, 12:141–157, December 1996.
- [RP81] George P. Richardson. and Alexander L. Pugh. *Introduction to System Dynamics Modeling with DYNAMO*. MIT Cambridge, MA, USA., edition, 1981.
- [Sha05] A. M. Sharif. Industrial Viewpoint can systems dynamics be effective in modeling dynamic business systems? *Business Process Management Journal*, 11(3):612–615, 2005.
- [vBFvdW97] P. van Bommel, P.J.M. Frederiks, and T.P. van de Weide. Object-Oriented Modeling based on Logbooks. . *The Computer Journal*, 39(9):793–799, February 1997.
- [Whi04] S.A. White. Business Process Modeling Notation (BPMN) Version 1.0. *BPML Org*, May 2004.

Application Landscape Metrics: Overview, Classification, and Practical Usage

Jan Stefan Addicks, Philipp Gringel

Software Engineering for Business Information Systems
OFFIS - Institute for Information Technology
Escherweg 2
26121 Oldenburg, Germany
{jan.stefan.addicks, philipp.gringel}@offis.de

Abstract: Due to mergers and acquisitions as well as uncoordinated projects, application landscapes of today's organizations contain redundant applications (two or more applications that have similar functionality). To consolidate the application landscape, comparisons of applications have to be performed. Application landscape metrics are seen as an appropriate instrument for such comparisons. This contribution describes a method as well as a template to classify landscape metrics. In presenting a tool prototype, we provide a first glance of how to practically employ application landscape metrics.

1 Introduction

The everyday business of today's organizations is supported by numerous business applications that form an application landscape (denoted as *landscape*). Landscapes continuously grow in complexity, due to continuous adaptations in order to fulfill business requirements. This leads to an increasing number of heterogeneous interwoven applications [Add09]. Since applications support business processes, they are also affected by alignment to changing customer markets or new market situations.

In order to improve the quality of landscapes, adequate documentation is essential. Stakeholders (like chief information officers (CIOs)) being in charge of the landscapes rely on documentation to manage current and plan future states. Scientific approaches often focus on modeling and documenting the landscape and the interconnections with hardware infrastructure and business elements (cf. [Fra02], [Lan05] and [WS06]). These results are complemented by commercial tools to support these issues. An overview of major tools can be found in [MBLS08], [Jam08] and [Pey07]. Since landscapes are part of enterprise architectures (EA), these tools typically address EA as a whole.

Nevertheless, in disciplines like IT consolidation (cf. [Kel07]), a well documented landscape alone is still insufficient to adequately support stakeholders. When two or more applications with similar functionality have to be compared to each other in order to determine the one that best suits the landscape, additional instruments are necessary. Land-

scape metrics as well as assessment and evaluation methods for landscapes seem suitable here. The mentioned techniques generate comparable values for each application under investigation. Based on the results, stakeholders can decide which application is to be removed from the landscape. So far, no established metrics-centered methods exist that allow for flexible as well as multiple usage of application landscape metrics (denoted as *landscape metrics* in the following), but several approaches address similar issues (cf. [Dur06], [Gam07], [GSL07], [Lan08], or [LS08]).

These approaches differ from classical software metrics (cf. [FP98] or [KKC00]), since the latter are used to evaluate the quality of single software systems within the software engineering phase and explicitly do not regard the environment in an organizational context. However, this aspect is important for the evaluation of an application within a landscape, since the properties of one application can influence characteristics of another (cf. [Add09]). For instance, the response time of an application can depend on the response time of another. Besides application-to-application dependencies, the quality classification of applications can further depend on other aspects, like strategic decisions or regulations from authorities (i.e. SOX (Sarbanes-Oxley Act) compliance or Basel II compliance [Kel07]).

To this point, there are neither fixed definitions of metrics for these concerns in general nor common overviews of landscape metrics. In order to present an overview of the current situation regarding landscape metrics we have performed a literature analysis. The objective was to identify landscape metrics as well as key figures that can be used to evaluate and benchmark applications regarding their landscape context. The identified metrics and key figures were classified using a method and a classification template which are both presented in this contribution. The resulting overview of existing landscape metrics provides a foundation for practitioners to choose adequate metrics to evaluate their landscapes.

When metrics are established in organizations, resulting key figures could effectively be used for monitoring purposes and benchmarking. Since software map visualizations [Wit07] are commonly used to provide overviews of landscapes, key figures can be depicted in such maps by small icons next to the symbol representing an application to indicate the value of an application's property, e.g. its quality. Tools providing this functionality exist (cf. planning IT from alfabet); however, those are in general based on proprietary models and metrics that often can not be customized easily by the user.

Tools that support various metrics and allow for integrating individual metrics have not come into the market so far. Thus, stakeholders are in general not able to define metrics on their own and use them for evaluation purposes. In order to motivate the practical usage of landscape metrics, we present a software prototype, which allows for flexible usage of metrics and various kinds of representations for the results.

The remainder of this paper is structured as follows: The next section presents a foundation of key figures and metrics and takes a quick glance at the state-of-the-art of landscape metrics. An overview of relevant research work is also given. Section 3 presents the research method, on which the results of this contribution are founded. We describe a method by which landscape metrics from different sources can be captured and compared to each other in a unified manner. The results of our literature research are summarized

in section 4. The usage of landscape metrics with our prototype is shown in section 5 before the last section sums up this contribution and gives an outlook on future research activities.

2 Metrics and Key Figures

We start this section by defining the central concepts of our contribution. According to [Küt09] a *key figure* captures facts quantitatively and in an aggregated manner, which means they are often used to map a large amount of data to few significant values. A key figure is the result of measuring a defined property using a concrete metric. A *metric* is a measuring specification (often a mathematical formula), that belongs to a property (e.g. the availability of an application). The metric defines in which way a resulting key figure has to be determined. Landscape metrics are used to evaluate applications' properties with regard to the dependencies within a landscape and thus are not limited to application-specific properties.

With regard to literature, the field of landscape metrics is rather young. There is only little literature available focusing on such aspects, but the number has continuously increased recently. Landscape metrics are intended to support CIOs by means of providing additional information of the applications' quality. However, there are only few case studies of metrics usage so far (mostly from academia like [LS08]).

This is in accordance with the results of a survey conducted by Lankes in 2007 (cf. [Lan08]). According to the survey, only 37% of the surveyed practitioners indicated that their organization actually uses metrics; however, 52% predicted a possible usage for the future. Only 11% of the surveyed practitioners regarded metrics as not applicable.

By means of presenting an overview of existing landscape metrics, we provide a foundation for practitioners to adapt these for their organization where appropriate. Nevertheless, there are further challenges to face. For instance, Lankes et al. [LS08], state the difficulty to convince CIOs to use metrics, because the difficulty in gathering the required data and correctly interpreting the metrics' results is perceived as an obstacle. The consumption of valuable time and resources for gathering the required data is seen as the main obstacle. Thus, the metrics' usage has to be as efficient and effective as possible, to provide more benefit to the people in charge.

Tool-support for the evaluation is additionally required to assist the process. Although there are tools that provide similar functionality, such as EAM (EA management) tools (cf. [MBLS08] for a survey of major tools), most of them cannot be extended or customized to meet individual requirements with respect to metrics.

We could not locate any comprehensive overview of metrics to evaluate landscapes. However single key figures and metrics suitable for evaluation of landscapes are presented and used ([Add09], [EHH⁺08], [SLJ⁺05], [JLNS07], [NJN07], [LS08], [Lan08], [SW05]). In other publications, further evaluation approaches can be found ([AS08], [Gam07], [JJSU07], [Wit07]). Some of the more sophisticated and recently released methods from academia will be described briefly below.

2.1 Failure Propagation

Five metrics for the calculation of failure propagation in applications and application landscapes can be found in [Lan08]. Results of the metrics' evaluation are presented in [LMP08]. The *testability* of an application is studied in [LS07].

Lankes [LS08] presents a proposal to preserve quality (e.g. regarding the availability of applications) in a landscape. To this purpose, the failure propagation within an application landscape is investigated. This investigation relies on the fact that connected applications are interdependent. An application relies on data and/or services provided by another application. This interwoven structure causes (even transitively) connected applications to operate incorrectly if an application fails or is behaving erroneously.

The presented approach also takes failure propagation within a single application into account. By using the software architecture, the interfaces of an application that are connected internally can be specified, so that the failure propagation from required interfaces to provided interfaces of an application can be evaluated. For this purpose, specific data as well as metrics that can make use of the data are required.

2.2 Business Value Assessment

In [Gam07], a method is introduced by which stakeholders of an enterprise can decide on alternatives (denoted as *System Scenarios*) in an upcoming IT investment. The enterprise is enabled to take a justified decision in favor of the system scenario which is capable of generating the greatest business value.

The method allows for a comprehensive examination and evaluation of an application landscape (i.e. the different system scenarios), including different points of view. Several assessments of each system scenario are obtained by people in charge, experts, and intended users.

Uncertain results from evaluations by humans are handled by the presented method. Furthermore, the percentage of fulfillment of functional and non-functional properties of a given scenario is stated. An additional range reflects the uncertainty and incompleteness in the experts' answers.

Such a method can be applied whenever experts' assessments are required in order to compensate for missing information.

2.3 Enterprise Architecture Analysis

Simonsson et al. ([SLJ⁺05]) present an approach to weigh different scenarios against each other and to decide in favor of the best solution available. According to [SLJ⁺05] the approach is cost-effective, easy to understand and scenario-based. On the basis of easily measurable system properties identified in literature, scenarios are compared to each other.

The presented approach provides an enterprise’s CIO with comprehensible information to support the decision making process.

In order to balance investment alternatives, the analysis of the different systems’ qualities is emphasized. Desirable quality attributes are, amongst others, availability, safety, functional suitability or interoperability. In [JJSU07], a software tool is described that supports the user in the creation of enterprise architecture models and their analysis regarding the desired quality attributes. Several quality attributes as well as some metrics that are relevant to the analysis of a system’s quality are mentioned in [JLNS07] and [NJN07].

3 Approach for classifying metrics and key figures

To achieve a categorization of key figures and metrics we introduce a method which allows for a comparison of different key figures. To provide a structured overview of key figures and metrics, for each key figure a table is created, providing an overview of the most relevant data. The table is similar to the key figure template introduced in [Küt09]. An example is presented in Table 1.

| Key Figure #00 | | 1.0(1) | [x,y] | ÄÄ |
|----------------------|---|--------|-------|----|
| Name | Name of the key figure. | | | |
| Description | A short description of the key figure. | | | |
| Use | What is assessed by the key figure? | | | |
| Required Data | Which kind of data is required to determine the key figure by a metric? | | | |
| Metric | The metric used to calculate the key figure’s value. | | | |

Table 1: Template for the key figure overview table

This table contains relevant information about a key figure. First of all, the name and a brief description are given. Whenever both those entries are not sufficient to explain the key figure’s usage, the additional field “use” can be filled in to provide further information. Since every key figure is based on concrete data, this information is also part of the table, in the row labeled “required data”. CIOs or other stakeholders may see with a quick glance which data has to be acquired to use the key figure.

This is quite an important aspect, since information gathering usually is time-consuming and thus expensive. Organizations in general do not have all required information at their disposal. It may either not have been collected at all or may be outdated. Thus, the objective to use a key figure might be connected with some effort which is made evident by the key figure overview table.

The last textual information in the table is a (formal) description of the underlying metric that has to be applied to compute the key figure’s value. Equation 1 presents an exemplary metric (metric for key figure *service availability*). The variables in the metrics have to be explained clearly in the “required data” section (cf. Table 3).

$$sA(oi) = \sum_{j \in I(oi)} A^{|j|} (1 - A)^{\#app - |j|} \quad (1)$$

Categorizations of key figures are represented by a code located in the upper right corner of the table. That code contains three sections separated by '|': *effort*, *way of comparison of results*, and *evaluation context*. These elements are explained in detail in the following.

Effort

The first element indicates how much effort is required to get the key figure's value. The lower the number, the smaller the effort. On the basis of the data required for the computation of the key figure's value, experts ought to estimate the necessary effort.

To determine this, the following steps have to be performed: For each entry in the required data field (cf. Table 1) the data volume has to be identified and assigned to one of the following categorization classes: *small*, *medium*, or *large*.

Moreover, the complexity of gathering the required data has to be figured out. The complexity can be assigned to either *easy*, *medium*, or *hard*. If the amount of data can be gathered automatically, the complexity is ranked as *easy*. There might be tools in the organization that hold the required data and no further manual processes to collect information have to be performed. For instance data about the *availability* of applications could be required, which is continuously checked by a monitoring tool. The complexity is regarded as *medium* whenever manual interventions are needed, for instance when automatically collected data has to be checked and filtered manually. If the required data is not available in digital form at all and it has to be collected manually, we define the complexity to be *hard*.

There are nine possible combinations of data volume and complexity of data collection that describe the effort (cf. Table 2). These combinations are named as (effort) *classes*. We define five different effort levels from 1 to 5, where 5 represents the greatest effort. Each effort class is assigned to an effort level. Table 2 visualizes these mappings.

| Class | Data volume | Complexity | | Mapping |
|-------|-------------|------------|---|---------|
| C#1 | small | easy | → | 1 |
| C#2 | small | medium | → | 1 |
| C#3 | small | hard | → | 3 |
| C#4 | medium | easy | → | 1 |
| C#5 | medium | medium | → | 2 |
| C#6 | medium | hard | → | 4 |
| C#7 | large | easy | → | 2 |
| C#8 | large | medium | → | 4 |
| C#9 | large | hard | → | 5 |

Table 2: Mapping of possible combinations of data volume and complexity on numbers.

The cumulative effort which has to be carried out to get all required data for the key figure is the result of the arithmetic mean of the classes assigned to each entry in the "required data" field. This result is rounded to one decimal place.

Although the average effort value might indicate a moderate effort, there could be some required pieces for which data acquisition effort is particularly high. To express such peaks, a second key figure can be assigned to the effort level. This key figure represents

the highest effort level for all sorts of required data and is shown in round brackets, next to the mean value. Hence an effort of 2.0(5) might imply more difficulties in data acquisition than an effort of 2.0(3).

Comparison of results

The second code element describes how results for the respective key figure can be compared. If the key figure is a number, the result range can be split into intervals. That is depicted by the term $[x, y]$. For each key figure the variety of assessment intervals has to be predefined, that is the number of intervals as well as their boundaries.

If a key figure's value is an element of a set of discrete concepts, the respective code contains the term $\{x_1, \dots, x_n\}$. Similar to the classification into assessment intervals, the discrete result sets have to be predefined. According to the definition of a key figure ([Küt09]) they have to be measurable on a scale. Thus, only elements within the defined set are valid and each element of the set is mapped to a number (for additional information see [Gri09]).

Evaluation context

One of the three symbols \mathbb{A} , \mathbb{L} and \mathbb{A} appears in the third and last element of the code. The symbol \mathbb{A} represents key figures that can be used for context-dependent evaluations of applications. Key figures that describe landscape properties are illustrated by a \mathbb{L} . The third group (depicted by a \mathbb{A}) contains the key figures that are targeting application properties. The key figures of the last two groups (\mathbb{L} and \mathbb{A}) do not represent indicators that consider the actual context of applications, that is for instance respective landscape with all the interdependencies to connected applications (cf. [AS08], [Add09]). However, the key figures of all three groups will provide valuable information about the landscape to the CIO.

4 Results

Performing the literature research, we identified 64 key figures in the investigated literature (cf. [Add09], [EHH⁺08], [JLNS07], [Lan08], [NJN07], [SLJ⁺05], [SW05], [Wit07]). These were classified using the proposed methodology (cf. section 3). Table 3 presents the key figure *serviceAvailability* ([Lan08]) as a concrete example that is deemed representative of the others.

In addition to the examination of the key figures we also took a closer look at the required data. Altogether 108 different types of data are needed to calculate the 64 key figures' values (after removal of duplicates).

| Key Figure #1 | | 2,3(4) | [x,y] | AL |
|---------------|---|--------|-------|----|
| Name | serviceAvailability (sA) | | | |
| Description | The key figure indicates the percentage of cases in which an interface <i>oi</i> is available. A "case" is a state of an application landscape (cf. [Lan08]). | | | |
| Use | - | | | |
| Required Data | <ul style="list-style-type: none"> • The set $I(oi)$, containing the states of an application landscape in which the respective interface is available. [C#3, C#6] • The availability A of the interface's <i>oi</i> application. [C#1] • The set $\#app$ of applications that are part of the application landscape under investigation. [C#1] | | | |
| Metric | $sA(oi) = \sum_{j \in I(oi)} A^{ j } (1 - A)^{\#app - j }.$ | | | |

Table 3: Key Figure #1: serviceAvailability

The data can be roughly divided into two main categories, *numeric*, i.e. countable and *categoric*, i.e. non-countable data. Numeric data is data like the *total number of applications in a landscape*. The categoric category summarizes data like the *software architecture of an application*. The group of numeric data contains 71 types of data (66%). 37 types of data (34%) relate to the categoric category. These two main categories were further refined. Each of them is divided into four sub-categories: *Intra-A/AL*, *Environment-A/AL*, *Key Figures* and *Detached Data* (cf. Figure 1). The diagram shows the distribution percentage of all data over the four sub-categories.

The group Intra-A/AL contains data that represents information about an application (A) or a landscape (AL), for instance the *number of modules per system*. This group accounts for a share of 55% of all data, numeric and categoric. Data that represents information about the operating environment of an application or an application landscape (e.g. *number of users of an application*) is collected in the group Environment-A/AL, which accounts for a share of 21%. If required data itself is again a key figure, such as the *coupling of pair wise considered applications*, it is assigned to the appropriate category, which accounts for a share of 9%. The remaining 15% of the data focuses on data that can be collected without taking a specific application or a landscape into account. Therefore it is summarized in the category Detached Data. Some kinds of arising costs are collected in this group. Altogether there are eight categories. The exact distribution of the data into these categories is presented and explained in [Gri09].

The effort estimation introduced in section 3 allows for the comparison of different key figures. If the data types required for calculating a key figure are known, it is easy to determine the effort by using the presented approach. The calculation of the effort is based on data types and not on actual data. That is why an instance-independent value is generated. Together with the suitability of a key figure for the evaluation of applications in the context of their landscape a simple way of comparing key figures is achieved. Figure 2 depicts the key figure distribution of the two categories effort and suitability for application landscape assessment.

The effort scale ranges from 1 to 5. For 41% of all key figures, the respective effort is

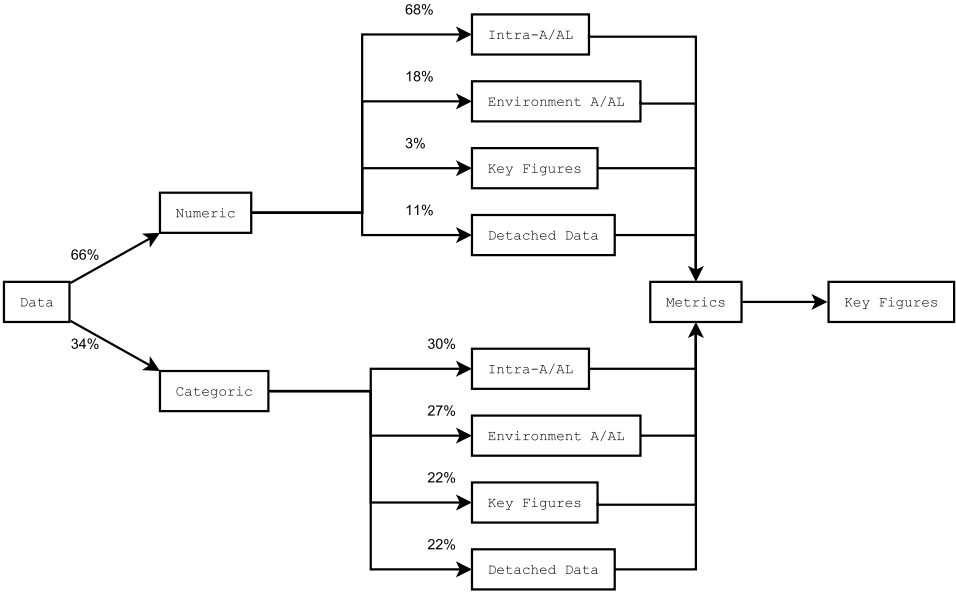


Figure 1: Distribution of all data on the different categories and sub-categories.

less than 1.5 (cf. Figure 2, left side). These key figures can be calculated easily and relatively quickly. This is of interest, if a quick overview of the application landscape under investigation is desired. Nevertheless, the given number may vary from organization to organization due to the heterogeneity of existing data. The diagram on the right in Figure 2 shows how many key figures are suitable for context-dependent evaluation of application landscapes (8%) and how many key figures are appropriate for assessing landscape properties (53%) and application properties (39%) respectively.

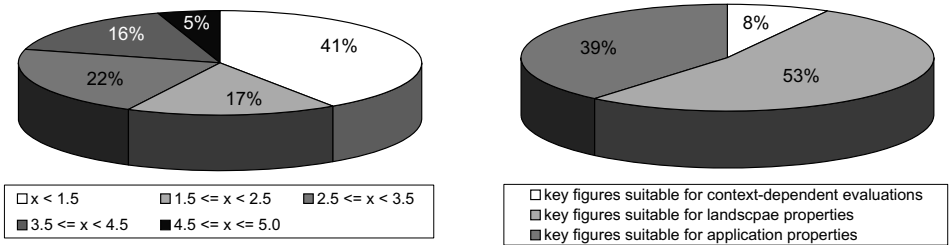


Figure 2: Distribution of effort (left) and suitability for application landscape assessment (right).

Due to the limited space of this contribution, we can only take a quick glance at our results and refer to [Gri09] which contains more detailed explanations as well as categorizations of all 64 key figures and further diagrams.

For the key figures' computation, the required data not only has to be collected but also its

topicality as well as correctness have to be assured. This is achieved through a continuous maintenance of data. The provision of necessary data is one of the largest challenges. To meet this challenge and the demand for faster and easier computation of key figures it is desirable to have this task supported by a software tool.

A central EA management cannot be assumed for all companies and thus not all relevant data is available quickly. Frequently crucial data is clustered in an employee's individual files, or is not available in digital or even written form (expert knowledge). In some situations it is not (yet) possible to do without the opinion of experts.

In order to draw the right conclusions from an application landscape's evaluation, an analysis should be done to identify the key figures to be used. Then the recommended key figures are computed. Despite the previous analysis, a subsequent interpretation of the results by experts is essential.

5 Usage of Application Landscape Metrics

To this stage no tools are available, which allow for a flexible usage of landscape metrics to evaluate an application landscape and thus support IT management processes. Therefore, we implemented a "lightweight" assessment tool which enables the inclusion of various kinds of metrics as a prototype. The tool relies on the Eclipse Rich Client Platform and thus allows for the inclusion of plugins to extend the software. To be highly flexible, we have built

- plugins to represent metrics,
- plugins to represent results of the evaluations, and
- plugins to contain the data model.

The first kind of plugin allows us to add, remove or modify metrics without changing the software itself. The second kind allows for exchanging the representation of the results, which are constituted of the applications examined and of the key figures of the selected metrics for each application. Thus, it is possible to get a tabular representation as well as a graphical representation of the results.

Figure 3 shows a screenshot of our prototype. While the left area of the screen contains a list of all applications and a list of all integrated metrics, the right area contains the elements of a landscape evaluation. In that area, the CIO has the opportunity to select the metrics to use in the evaluation as well as the applications to be evaluated. The latter aspect is based on the selection of a certain landscape view. After executing the evaluation by clicking the respective button, the results are displayed in the bottom area. The kind of visualization depends on the visualization plugins, which are loaded in the starting phase of the software. The screen in Figure 3 shows a graphical representation. The representation depicts all applications as squares containing the application's name. All selected metrics are depicted as symbols (in this example crosses, rectangles, and circles) with the resulting key figure next to the corresponding application.



Figure 3: Screenshot of our metrics tool-prototype

This visualization is intended to support CIOs at architectural decisions by presenting the quality of applications regarding the selected metrics. Since metrics are defined as plugins, these can easily be added or removed, without modification of the tool itself. All included metrics are listed in the metrics overview window. Beside the metrics' names, the list also contains the complexity classifications (cf. section 3), the key figure objectives (cf. Figure 2), and the classification of the key figure with regard to the taxonomy described in section 3. All this information must be specified in a proprietary XML-file as part of the plugins. In future versions of the software this list is intended to also present the percentage of required data so that the CIO can decide not to use a metric, if the required data is unavailable.

We prototypically implemented three concrete metrics: availability (cf. [Gri09]), technology heterogeneity (cf. [Gri09]), and BIO (business information object) criticality (cf. [Add09]). The first and the last metric require information about the landscape structure, since the metrics' definitions use the connections between applications for the computation; the second one compares the technology of an application with the technology of the other applications. Detailed explanations are presented in [Gri09]. These metrics are landscape metrics since they cannot be applied to a single application without taking into account further artifacts.

6 Conclusion and Outlook

We have presented a categorization method for key figures resulting from landscape metrics. The categorization gives an overview of them and makes them comparable regarding the effort of collecting the required data. The structure of the key figure presentation allows for estimation of the trade off between data gathering effort and benefit of the key figures' values.

This categorization method was applied to the results of a literature research that aimed at identifying metrics and key figures which can be used to evaluate landscapes and applications regarding their EA context, for instance their dependencies to other artifacts. 64 key figures were identified in literature. To apply the metrics in order to get the key figures' values, data is required. Along with the results, 108 different types of data were identified and presented in the key figure overview tables (cf. example in Table 3).

To give a brief demonstration of how to use the various metrics and key figures, we have presented a prototypical tool which uses a modular architecture to provide extensibility towards metrics, the underlying data model, and the result presentations.

One plan for future activities is to refine and improve the categorization by gathering more knowledge about the actual amount and the actual complexity of required data. The effort's estimation may rely on empirical values instead of experts' estimations. In order to receive this information, a survey has to be conducted aiming at organizations that already make use of landscape metrics.

In the example scenario we have applied three metrics, thus we have received three key figure values for each application. In order to compare two applications each key figure has to be compared to the according one of the other application. The more metrics are used the more complex the comparisons will get. Additionally the metrics may have different weighting factors that also have to be taken into account. To improve the usage with respect to this aspect, a method to aggregate key figures to receive a single summarizing indicator for an application is required.

References

- [Add09] Jan Stefan Addicks. Enterprise Architecture Dependent Application Evaluations. In *Proceedings of the 3rd IEEE International Conference on Digital Ecosystems and Technologies (IEEE Dest 2009): Cyber Engineering and Creating Value by Making Connections*, Istanbul, 2009.
- [AS08] Jan Stefan Addicks and Ulrike Steffens. Supporting Landscape Dependent Evaluation of Enterprise Applications. In Martin Bichler, Thomas Hess, Helmut Krcmar, Ulrike Lechner, Florian Matthes, Arnold Picot, Benjamin Speitkamp, and Petra Wolf, editors, *Multikonferenz Wirtschaftsinformatik, MKWI 2008, Proceedings: [26. - 28. Februar 2008, TU München in Garching]*, pages 1815–1825. GITO-Verlag, Berlin, Berlin, 2008.
- [Dur06] Michael Durst. Kennzahlengestütztes Management von IT-Architekturen. In Hans P. Fröschle and Susanne Strahringer, editors, *IT-Governance*, volume 250 of *HMD*, pages 37–48. dpunkt-Verl., Heidelberg, 2006.

- [EHH⁺08] Gregor Engels, Andreas Hess, Bernhard Humm, Oliver Juwig, Marc Lohmann, Jan-Peter Richter, Markus Voß, and Johannes Willkomm. *Quasar enterprise: Anwendungslandschaften serviceorientiert gestalten*. dpunkt.verlag GmbH, Heidelberg, 1. aufl. edition, 2008.
- [FP98] Norman E. Fenton and Shari Lawrence Pfleeger. *Software metrics: A rigorous and practical approach*. PWS Publ., Boston, 2 edition, 1998.
- [Fra02] Ulrich Frank. Multi-perspective enterprise modeling (MEMO) conceptual framework and modeling languages. *Proceedings of the 35th Annual Hawaii International Conference on System Sciences, 2002. HICSS*, pages 1258–1267, 2002.
- [Gam07] Magnus Gammelgaard. *Business Value Assessment of IT Investments: An Evaluation Method Applied to the Electrical Power Industry: Dissertation*. PhD thesis, Royal Institute of Technology (KTH), Stockholm, 2007.
- [Gri09] Philipp Gringel. *Metriken zur Bewertung von Anwendungslandschaften*. Diplomarbeit, Carl von Ossietzky Universität, Oldenburg, 2009.
- [GSL07] Magnus Gammelgaard, Marten Simonsson, and Åsa Lindström. An IT management Assessment Framework: Evaluating Enterprise Architecture Scenarios. *Information Systems and E-Business Management*, 5(4):415–435, 2007.
- [Jam08] Greta A. James. Magic Quadrant for Enterprise Architecture Tools, 2008.
- [JJSU07] Pontus Johnson, Erik Johansson, Teodor Sommestad, and Johan Ullberg. A Tool for Enterprise Architecture Analysis. In *Proceedings of the 11th IEEE International Enterprise Computing Conference (EDOC 2007)*, October 2007.
- [JLNS07] Pontus Johnson, Robert Lagerström, Per Närman, and Mårten Simonsson. Extended Influence Diagrams for System Quality Analysis. *Journal of Software*, September 2007.
- [Kel07] Wolfgang Keller. *IT-Unternehmensarchitektur: Von der Geschäftsstrategie zur optimalen IT-Unterstützung*, volume 1. dpunkt.Verl., Heidelberg, Neckar, 1. aufl. edition, 2007.
- [KKC00] Rick Kazman, M. Klein, and Paul C. Clements. ATAM: Method for Architecture Evaluation, 2000.
- [Küt09] Martin Kütz. *Kennzahlen in der IT: Werkzeuge für Controlling und Management*. dpunkt.verlag GmbH, Heidelberg, 3 edition, 2009.
- [Lan05] Marc M. Lankhorst. *Enterprise Architecture at Work: Modelling, communication, and analysis*. Springer, Berlin, 2005.
- [Lan08] Josef Lankes. *Metrics for Application Landscapes: Status Quo, Development, and a Case Study: Dissertation*. PhD thesis, Technische Universität München, München, 2008.
- [LMP08] Josef Lankes, Florian Matthes, and Tarmo Ploom. Evaluating Failure Propagation in the Application Landscape of a Large Bank. *Comparch 2008 Industrial Experience Report Track, Karlsruhe*, 2008.
- [LS07] Josef Lankes and Christian M. Schweda. Constructing Application Landscape Metrics: Why & How. Garching b. München, 2007.

- [LS08] Josef Lankes and Christian M. Schweda. Using Metrics to Evaluate Failure Propagation and Failure Impacts in Application Landscapes. In Martin Bichler, Thomas Hess, Helmut Krcmar, Ulrike Lechner, Florian Matthes, Arnold Picot, Benjamin Speitkamp, and Petra Wolf, editors, *Multikonferenz Wirtschaftsinformatik, MKWI 2008, Proceedings: [26. - 28. Februar 2008, TU München in Garching]*, pages 1827–1838. GITO–Verlag, Berlin, Berlin, 2008.
- [MBLS08] Florian Matthes, Sabine Buckl, Jana Leitel, and Christian M. Schweda. Enterprise Architecture Management Tool Survey 2008, 2008. Technische Universität München, Chair for Informatics 19 (sebis).
- [NJN07] Per Närman, Pontus Johnson, and Lars Nordström. Enterprise Architecture: A Framework Supporting System Quality Analysis. In *Proceedings of the IEEE International Annual Enterprise Distributed Object Computing Conference (EDOC)*, volume 11, October 2007.
- [Pey07] Henry Peyret. The Forrester Wave: Enterprise Architecture Tools, Q2 2007, (<http://www.forrester.com/go?docid=41915>), 2007.
- [SLJ⁺05] Mårten Simonsson, Åsa Lindström, Pontus Johnson, Lars Nordström, John Grundbäck, and Olof Wijnblad. Scenario-Based Evaluation of Enterprise Architecture - A Top-Down Approach for CIO Decision-Making. In *Proceedings of the International Conference on Enterprise Information Systems*, May 2005.
- [SW05] Alexander Schwinn and Robert Winter. Entwicklung von Zielen und Messgrößen zur Steuerung der Applikationsintegration. In Otto K. Ferstl, Sven Eckert, Tilman Isselhorst, and Elmar J. Sinz, editors, *Wirtschaftsinformatik 2005: EEconomy, eGovernment, eSociety*, Springer-11775 [Dig. Serial], pages 587–606. Physica-Verlag Heidelberg, Heidelberg, 2005.
- [Wit07] André Wittenburg. *Softwarekartographie: Modelle und Methoden zur systematischen Visualisierung von Anwendungslandschaften*. Dissertation, Technische Universität München, München, 2007.
- [WS06] Robert Winter and Joachim Schelp. Reference Modeling and Method Construction: A Design Science Perspective. In M. Lorie, editor, *Proceedings of the 21st Annual ACM Symposium on Applied Computing (SAC2006)*,. ACM Press New York, NY, USA, 2006.

How do system and enterprise architectures influence knowledge management in software projects? – An exploratory study of six software projects

Carsten Lucke, Markus May, Nane Kratzke, Ulrike Lechner

Institut für Informatik
Universität der Bundeswehr München, Germany
Werner-Heisenberg-Weg 39
85577 Neubiberg
carsten.lucke@unibw.de
markus.may@unibw.de
kratzke@iabg.de
ulrike.lechner@unibw.de

Abstract: This paper covers the influence of system- and enterprise architectures on knowledge management in software development projects. The common impact of architectures is researched in the context of six case studies of medium and large sized software- and system development as well as technical and organizational consultancy companies in the military and non-military domain. Observations are collected in a wide variety of aspects and evaluated on the basis of the Probst et al. knowledge management model [PRR06].

1 Introduction

A lot can happen in a several decades long lifespan of a major information system or enterprise: minor and major changes of its structure, requirements or technology. How to make sure that the knowledge about the requirements, the design decisions, algorithms or processes and the history of changes is sustained such that necessary changes can be made now and in the future? How to prepare for prospective changes?

Usually in complex projects, at some point the majority of the consultants, system designers and engineers who initially built a system or an enterprise architecture will have left the project or even the company. Nevertheless, emerging issues need to be checked and solved as fast as possible. Concerning software systems the arising questions could be: How severe is the reported software problem? Does its underlying cause lie in the software, the system design, the requirements, the handling of the system by users or in the training of the users? Considering enterprise architectures a possible issue could be a changed market situation to which the company needs to respond by establishing a new business strategy: Do certain business processes need to be changed? Is the IT infrastruc-

ture able to cope with intended changes?

These kinds of questions can only be properly answered, if the responsible people have access to knowledge: knowledge about the actual system or enterprise requirements, design decisions, algorithms, documentation and training material and this knowledge needs to be kept alive for decades to be able to respond to those issues. Our research questions are: In how far are system and enterprise architecture used for knowledge management? Who uses them? What kind of support do they provide for knowledge management and how does their usage shape the knowledge management? In how far are they embedded in knowledge management processes and systems?

In our exploratory approach, we present six short cases for usage of architectures in software projects. We collect best practices for system and enterprise architecture usage and the integration of knowledge management and system architectures. We conclude this exploratory study with several hypotheses, based on the identified best practices.

This paper is organized as follows. First, we briefly discuss relevant literature (Sect. 2) and present our method (Sect. 3). Afterwards, we present the six researched cases (Sect. 4), collect the best practices (Sect. 5) and conclude with the presentation of our hypotheses and a brief discussion (Sect. 6 + 7).

2 Theory

The IEEE-1471 standard describes architecture as *fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution [Hil00]*. The term architecture is defined heterogeneously throughout literature. Already in the 1990s authors mourned the proliferation of the term »architecture« [Krc90]. When talking about architecture in this paper, we refer to the theory of Armour et al. [AKL99b], depicting architecture as a pyramid: *A software architecture describes the layout of the software modules and the connections and relationships among them. A hardware architecture can describe how the hardware components are organized. However, both these definitions can apply to a single computer, a single information system, or a family of information systems. Thus »architecture« can have a range of meanings, goals, and abstraction levels, depending on who's speaking. An information system architecture typically encompasses an overview of the entire information system – including the software, hardware, and information architectures (the structure of the data that systems will use). In this sense, the information system architecture is a meta-architecture. An enterprise architecture is also a meta-architecture in that it comprises many information systems and their relationships (technical infrastructure). However, because it can also contain other views of an enterprise – including work, function, and information – it is at the highest level in the architecture pyramid. Enterprise architecture is also frequently referred to as system of systems [LS06]. Seen in that way it has to take into account that each system has its own environment of people, sub-*

systems, and data – plus each system must work with other systems to support business operations [AKL99a].

The Zachman framework [Zac87] may possibly be seen as the first major contribution which points out the importance of architectural descriptions as a way to handle increasingly complex (software) systems. Zachman has a strong focus on information systems. However, the proposed framework is nowadays even used as enterprise architecture framework and has inspired many of the currently available frameworks used for enterprise architecture. Most of them have incorporated the use of different perspectives/views to a system, as Zachman suggested. For instance the NATO Architecture Framework (NAF) defines seven views [Cou07] and The Open Group Architecture Framework (TOGAF) provides an architecture vision based on four views [Gro09]. The work of Schekkerman [Sch04a] gives a good overview on a lot of enterprise architecture frameworks and also shows the multiplicity of available solutions.

Several surveys conducted in the recent years show that especially enterprise architecture has grown to a strategic instrument of governance and business alignment [Sch05]. Another approach to standardize architectural descriptions is the IEEE-1471 standard [Hil00]. Maier et al. state that this standard *identifies sound practices to establish a framework and vocabulary for software architecture concepts* [MEH01]. But IEEE-1471 does not only focus on software-intensive systems but also on more general systems like information systems or systems of systems [MEH01]. Lankes et al. find that this standard is even useful for software-cartography of whole landscapes of systems [LMW05]. Referring to Schönherr architecture management and modeling experience a kind of renaissance in the field of enterprise application integration (EAI) since enterprise application frameworks have already been addressing different integration scenarios which bother IT-organizations today [Sch04b].

We observe that architectures are commonly used and broadly accepted in a variety of areas but their benefit to knowledge management, so far is hardly discussed up. In our opinion architectures support knowledge management in organizations, when being properly communicated and made accessible. In this exploratory research work we do neither focus nor constrain on a certain kind of knowledge like tacit or explicit [NT95] but try to gather whatever impact or influence architectures can have on knowledge management. In their reviewing assessment on knowledge management and knowledge management systems (KMS) Alavi and Leidner state that according to Davenport and Prusak [DP98] most knowledge management projects have one of three aims: *(1) to make knowledge visible and show the role of knowledge in an organization, mainly through maps, yellow pages, and hypertext tools; (2) to develop a knowledge-intensive culture by encouraging and aggregating behaviors such as knowledge sharing (as opposed to hoarding) and proactively seeking and offering knowledge; (3) to build a knowledge infrastructure – not only a technical system, but a web of connections among people given space, time, tools, and encouragement to interact and collaborate* [AL01]. We suppose that architectures can especially support (1) by providing access to architectural descriptions, which can possibly improve a variety of knowledge management processes.

3 Method

We follow a case-study approach researching six cases, which is considered a reasonable number of cases for a multi-case research [Eis89]. We inductively build our theory on how architectures can be used to support and improve knowledge management. We then identify best practices which emerge from the cross case analysis. The final results of our exploratory research are hypotheses.

Probst et al. define a model consisting of six core processes relevant for knowledge management [PRR06]: (a) knowledge identification, (b) knowledge acquisition, (c) knowledge development, (d) knowledge distribution, (e) knowledge utilization and (f) knowledge preservation. Since this model provides a practically relevant segmentation of knowledge management processes we use it to guide our research and reflect which processes benefit from the use of system or enterprise architectures.

The cases are compiled from interviews with senior system engineers, architects or project managers. The companies employing our interviewees are assembled from different backgrounds (i.e. business firms, government and defense area) to have a good variation and reach meaningful results. The interviews were semi-structured and the main topics covered in the interviews are the approaches in architecture design and management as well as the use of architecture in the daily work routines and the way architectures are communicated to different audiences. Four cases centre on system architecture and the other two on enterprise architecture. We chose to research these two architectural types since they are the most commonly used ones and to cover a range of differing approaches. Another reason is the probability to draw conclusions from the findings concerning one architectural type to the other one. All interviews were conducted in May and June 2008 in the context of a diploma thesis [May08]. We use the interview transcriptions and findings of this diploma thesis as the foundation of this research paper.

4 Cases

Our case studies cover projects in the following six organizations: Bayerische Motoren Werke Group (BMW), European Aeronautic Defence and Space Company (EADS), Industrieanlagen-Betriebsgesellschaft mbH (IABG), International Business Machines Corporation (IBM), Thales Group and the Taktikzentrum Marine (TZM) of the Marineoperationsschule (MOS) Bremerhaven. For anonymization purposes, these organizations are referred to as »Company A-F«, whereas the alphabetical order of these names does not allow for a reverse mapping.

Table 1 shows a summary of the effects of the companies' architecture on the knowledge management processes defined by Probst et al. [PRR06]. A tilde character (~) indicates no particular support, (+) means that a supportive character is perceived and (++) indicates significant support of the regarding knowledge management process.

| Knowledge Management Process | Company | | | | | |
|---------------------------------|---------|---|----|---|----|---|
| | A | B | C | D | E | F |
| <i>knowledge identification</i> | ++ | + | ++ | ~ | ++ | + |
| <i>knowledge acquisition</i> | ~ | ~ | + | ~ | + | ~ |
| <i>knowledge development</i> | + | + | + | + | + | + |
| <i>knowledge distribution</i> | + | ~ | + | + | ++ | ~ |
| <i>knowledge utilization</i> | + | + | + | + | + | + |
| <i>knowledge preservation</i> | ~ | + | + | + | + | + |

Table 1: Overview on how the architecture supports knowledge management processes

Table 2 gives an overview about the target audiences and the degree of utilization of the architecture for knowledge management. The tilde character (~) means no particular use of the architecture, (+) indicates noticeable but irregular use and (++) signalizes significant use of the architecture for knowledge management purposes. The audience classifications are *Manager* (project leader and project manager), *Architect* and *Developer* (with the architect being a chief developer), *Quality Assurance (QA) person* and *End-User*. Concerning system architectures the end-user is a person who uses the developed information system. Regarding enterprise architectures the end-user is a collective term that stands for the stakeholders defined in the enterprise architecture framework.

| Target Audience | Company | | | | | |
|---------------------------------|---------|----|----|---|----|----|
| | A | B | C | D | E | F |
| <i>Manager</i> | + | + | ++ | ~ | ++ | + |
| <i>Architect</i> | | ++ | ++ | | ++ | ++ |
| <i>Developer</i> | ++ | ++ | ++ | + | ++ | + |
| <i>Quality Assurance person</i> | + | ~ | + | ~ | ++ | + |
| <i>End-User</i> | + | ~ | ++ | ~ | ++ | ~ |

Table 2: Overview about target audiences of the architecture¹

Subsequently follows a more detailed description of the researched cases. We focus on the description of those aspects where architectures have shown to be supportive (+) or significantly supportive (++) for certain knowledge management processes (compare Table 1) or have noticeably (+) or significantly (++) been used by a target audience for knowledge management purposes (compare Table 2).

¹The role of architects was not explicitly mentioned in the interviews with Company A and D, thus the table-cells are left blank intentionally. It can be assumed that the value evaluated for developers applies to architects as well.

4.1 Company A

The project in Company A developed a proprietary multi-tier system architecture. This had a negative impact on the knowledge management processes of »knowledge utilization« and »knowledge acquisition« identified by Probst et al. [PRR06]. First, it was difficult to find employees being already familiar with this architecture and second, new project members generally needed some time to familiarize themselves with this kind of architecture. If the company would have adopted widely known architectures (i.e. open-source architectures) the probability of new project members knowing these would have been higher.

Furthermore, Company A maintained two different perspectives to the system architecture – one of these being data-related and another one providing a functional view to the system. The benefit of having these architectural descriptions was the development of a common vocabulary, used within communications between team members (management, developers and QA personnel). There was no dedicated perspective for the customer’s specialist division or end-users. Concerning project lead and project management the functional perspective was used, to define project teams and also documentation artifacts and configuration management were structured in regard to the functional perspective of the system architecture. This enabled project members having certain knowledge about the system architecture (mostly developers), to find relevant contact persons within the whole team, find relevant pieces of documentation in shorter time and navigate the configuration management system in a more efficient manner.

The software development process adopted by Company A was aligned according to the V-Model 97 [DW99] process model. In this regard, the biggest ascertainable advantage according to knowledge management was, that the process model postulated a certain amount of documentation artifacts to be created.

4.2 Company B

The case study of Company B explored a project concerned with system architecture creation. An external service provider developed the architecture for Company B. The specialist division of Company B was not interested in certain aspects of the architecture but only in the business needs that the final system was designed to fulfill as well as integration aspects of the system into the whole IT landscape. As a result the project maintained no dedicated perspective for employees of Company B. According to Zachman, a »model of the business (i.e. owner’s view)« would be a good perspective [Zac87] on how an information system works and it is well targeted at the audience of a business-specialist division. In this case though, no such perspective was documented. However, the external service-provider (which includes developers and architects) gained much advantage from properly documenting the scope of the architecture’s »technology model« as well as »out-of-context views« [Zac87]: The technology model includes data and process descriptions

and our interview partner mentioned that this model fostered a common vocabulary, used by the external service-provider's employees (managers, developers and architects). Various graphical user interface prototypes (out-of-context views) were used to support the communication with business-specialists as well as end-users.

It attracted our attention that project members of Company B structured their documentation artifacts in regard to the project stages (also in terms of directory structures in connection to file storage) whereas the service provider, who developed the information system architecture, organized its documentation deliverables in regard to the components of the information system architecture. In this regard, the architecture helped managers, architects, developers and QA personnel to structure documentation artifacts. Considering the Probst dimension of »knowledge identification« [PRR06] this means, the architecture adds certain value to identify relevant documentation artifacts. Having the team- and document structure aligned in regard to the architecture's components, helps to find contact persons or relevant documents. This way, it got possible to tell which documents needed to be updated, when specific modules of the system architecture were changed.

4.3 Company C

The project of Company C was about developing an enterprise architecture for a customer. Subsequently we use the term service provider synonymously when we refer to Company C. The enterprise architecture framework to be used – the NATO Architecture Framework (NAF) [Cou07] – was determined by the customer. The NAF defines several views to the architecture: e.g. an operational one and a system view. The operational view provides information about the organization's operational nodes, their tasks and how they are connected to each other. The system view illustrates the functionalities of the customer's organizational units. Describing the enterprise architecture with this set of different perspectives, allowed employees of the customer with different knowledge or backgrounds to get an easier and better understanding of the described circumstances.

In this case, no dedicated process model was used in the design process of the enterprise architecture model. Thus, the process model could not act as a catalyst for creation of documentation artifacts. However, this had no negative effect on the knowledge management since the enterprise architecture framework suggested a development method and defined certain architectural descriptions (views) to be created.

The process of analyzing the customer's current organizational architecture and the definition of the target enterprise architecture aligned to the business goals was active knowledge management. The employees of both, the customer and the service provider got deeply involved with the current enterprise architecture which allowed them to identify weaknesses (i.e. redundancies or disharmonies) and derive necessary changes for the architecture to be defined.

Mapping these facts to the Probst knowledge management model [PRR06], the architecture fosters a very effective »knowledge identification« which in turn directly supports »knowledge acquisition« and »knowledge development«. The creation and application of a clearly defined architectural model resulted in the establishment of a common vocabulary on both sides, customer and service-provider. This enhanced and alleviated communication inside and across teams. For this reasons, all target audiences (management, end-users, QA personnel as well as architects and developers) were assigned a (+) or (++) rating in table 2.

As a final result of the project, Company C defined and established an enterprise architecture, which could be presented to the customer's employees in different views. These different models were enriched with documentation artifacts providing detailed information. Hyperlinking all these elements allowed for an efficient navigation and combined a good overall overview with easy comprehensibility. Thus, the aspect of »knowledge identification« [PRR06] is strongly supported by the established enterprise architecture.

4.4 Company D

In the case of Company D a proprietary system architecture was developed, based on established architectural styles like client/server and peer-to-peer. While the usage of a proprietary architecture lowered the possibility of finding staff being familiar with it, the well known architectural styles partly compensated this downside and reduced the amount of knowledge new members of the project had to acquire.

The company maintained only a single perspective to the system architecture which was – referring to Zachman [Zac87] – a »technology model« providing »data- and process descriptions« of the system. This lack of different perspectives to the system architecture excluded project members, apart from developers and architects, from getting acquainted with the architecture. It is perceived to be unfortunate that even the model's target audience (i.e. software engineers and -architects) did not capaciously use the model, as it was outdated since changes to the physical architecture had not been incorporated in the documents. The communication about different aspects of the software system within specialized teams (e.g. developer to developer communication) worked rather well. But it was apparent, that as soon as communication between project teams or team members with different backgrounds (e.g. software-engineer to end-user communication) was required, the lack of a shared vocabulary or awareness of the system in general prevented those team members from communicating efficiently.

4.5 Company E

In the case of Company E an enterprise architecture was developed for a customer, based on the TOGAF framework [Gro09]. The framework defines four architectural domains:

business architecture, application architecture, data architecture and technical architecture. Company E added the dimension of »capability architecture« to the framework. The purpose of this additional dimension was to allow for a periodically conducted analysis of the companies capabilities.

The goal of the creation-process of the new enterprise architecture was to provide all knowledge regarding the five dimensions (e.g. detailed information about business units, IT infrastructure landscape, etc.) within an architecture database, to make information searchable. Several facts are apparent in this case: Already the analysis of the customer's current enterprise architecture adds major value in terms of »knowledge identification«. Since members of all audiences listed in table 2 were involved in this process they all benefited in a positive way. The creation of an architecture database containing valuable information about the enterprise architecture combined with powerful search functionalities helps to identify relevant knowledge and at the same time »knowledge distribution« gets a significant boost. Knowledge retrieval was changed from a »push« to a »pull« principle which reduced information overload. Information inside this architecture database was kept up-to-date with the support of specific tools.

The »knowledge utilization« dimension defined by Probst et al. [PRR06] is supported by providing information in different perspectives, each adjusted to a specific target audience. This supports ease-of-use of the available information. Employees on a strategic level (i.e. management) for instance, get an overview about the enterprise as a whole and processes whereas IT personnel can view information from an application- or technical infrastructure centric perspective.

Last but not least »knowledge acquisition« and »knowledge development« are improved by introducing the additional architectural domain of abilities, which allows for a periodical analysis on whether knowledge needs to be acquired externally or developed internally in order to accomplish enterprise goals.

4.6 Company F

The fundamental observation in the case concerned with Company F is, that lack of time resulted in a significant shortage of documentation. Focus of the project was the development of a client/server-oriented system architecture for a customer, using a proprietary architecture framework. Nevertheless, the fact that the well-known client/server architecture model was used, improved chances, that new project members (in most cases software engineers) were familiar with the very basics of the system architecture. Thus, considering the Probst et al. dimension of »knowledge acquisition« [PRR06] it is helpful to stick to non-proprietary products.

Since only a »technology model« perspective of the system architecture was maintained, the model was neither used for communication towards the customer, nor did it enhance

the building of a common vocabulary that could be used in conversations between employees of Company F and the customer. The customer was only interested in the fulfillment of its requirements and showed no interest in the architecture. As described in previous cases, team structure and also the structure and outlining of documentation artifacts were aligned to the technology perspective of the system architecture. Having a certain knowledge of the system architecture, this eased »knowledge identification« (i.e. finding appropriate contact persons or finding relevant documents).

The project employed the V-Model 97 [DW99] process model during the software development process. This process model defined necessary documentation artifacts to be created and thus did add a certain value in terms of »knowledge preservation«.

5 Best Practices

The six cases cover four companies developing system architectures and two companies developing enterprise architectures. Several best practices are gathered from these cases. One obvious best practice is to *incorporate a well-known architecture* rather than a not well-known one (e.g. a proprietary architecture) to support »knowledge utilization«. This best practice is independent from the type of architecture being created – enterprise or system architecture. Companies A, D and F dealt with proprietary system architectures which made »knowledge acquisition« and »knowledge usage« difficult. Still, Company D and Company F had an advantage over Company A by using well-known technology models like client/server or peer-2-peer in their system architecture framework. In these companies new project members were at least familiar with the underlying technologies of the architecture.

Another best practice is to *define architecture-models, showing the architecture from different perspectives*, each targeted at a specific audience. Both, »knowledge utilization« and »knowledge development« benefit from this practice. The cases showed, that independent from which target audience is addressed, an architectural model is rather used, if it is understandable for the specific group of people, than a model, which is too detailed or even outdated (Company D). Whenever a model is accepted and used by the target audience, it helps to establish a common vocabulary, which makes communication easier and more efficient (Company A, B, C and F). Cases concerned with system architecture development typically used architectural models and related vocabulary for communication within development teams, but not for communication towards end-users or customers (Company A, B and F). By contrast, all cases concerned with enterprise architecture, developed much more architectural models to give a view on the architecture from several different perspectives. In each of these cases, this resulted in the establishment of a commonly used vocabulary, enhancing communication within teams and across team- or division-boundaries (Company C and E).

One important lesson learned from the cases researching system architecture is to *structure*

documentation artifacts, configuration management systems (e.g. bug-tracking systems, source-code management systems) or even teams in regard to the architecture. Project members, having a basic knowledge about the architecture, benefit from this by finding relevant information or contact persons in an efficient manner (Company A, B and F). This practice supports the »knowledge identification« dimension identified by Probst et al. [PRR06]. This structuring helped Company B in terms of »knowledge preservation«, since whenever changes were done to the physical architecture it was obvious which documentation artifacts needed to be updated.

The cases, which dealt with enterprise architectures (Company C and E), did also focus on the structuring of knowledge management artifacts (e.g. documents, contact person information) in alignment to the architecture. Even better, beyond providing a plain architectural description, they did *use the architecture model(s) as interactive index of contents*. We call this a knowledge map. Models, depicting structural components of the enterprise architecture, are made accessible to employees in different views, showing the architecture from different perspectives. This gives »knowledge identification« as well as »knowledge utilization« a significant boost. Furthermore, Company C enriched these models by hyper-linking documents or other knowledge management artifacts, providing detailed context information about certain nodes in these architectural models.

6 Research Hypotheses

This section of the paper has a reflective look at a selection of our most important hypotheses and determines whether the observed results corroborate them.

1. *Both, system and enterprise architecture have an influence on the knowledge management processes defined by Probst et al. [PRR06].*

This assumption proves to be true in several aspects and the influence can be positive as well as negative. Company A for instance used a proprietary system architecture framework and had difficulties finding new project members, being already familiar with the architecture. Thus, the architecture has a negative influence on »knowledge acquisition«. On the other hand the cases show that architectural descriptions can support a number of knowledge management processes like for instance »knowledge identification« (Company C and E), »knowledge distribution« (Company E) or »knowledge preservation« (Company B).

2. *The use of an architecture leads to the establishment of a common vocabulary and therefore improves communication between team members.*

In all case studies, a common vocabulary is introduced by having an architecture or more precise by having one or more architectural descriptions. System architectures mostly help to build a common vocabulary within teams (Company A, B, D and F), whereas enterprise architectures help to introduce a vocabulary used within

teams as well as across team boundaries (Company C and E). An ideal architecture offers an architectural description to each target audience it intends to address. Enterprise architecture seems to be very effective in reaching its intended target audiences (see table 2). Certainly the reason for this is, that an architectural description is developed for each of the stakeholders and the defined target audience does not go beyond this group of people. However, typically a company consists of many more people but those stakeholders. We believe that extending the use of an enterprise architecture to a wider audience would be to the further benefit of a company's knowledge management processes. In any case, the architectural descriptions help to make communication more efficient and avoid misunderstandings.

3. *An architecture will not be used by people who do not understand the architectural descriptions.*

This hypothesis is closely connected to hypothesis (2). Most cases, concerned with both, system and enterprise architecture, provided more than a single perspective/view to the architecture. In the case of Company B for instance, the external service provider used a »technology model« [Zac87] to enhance the communication between software engineers (inside-team communication) and »out-of-context views« [Zac87] to enable and improve the communication between software engineers and business specialists or end-users (cross-team communication). On the contrary, Company A did not provide an architectural description suited to the business specialist division of the customer. In the result the architecture was not used in communications to the customer (across-team communication). Thus, it did only enhance the communication within the team of Company A. This tells us, that providing more perspectives/views to an architecture, means getting through to a wider target audience. In this regard architecture frameworks need to find a reasonable number of architectural descriptions.

4. *Architectural descriptions are perfectly suited to be used as knowledge maps and will as such improve several of the knowledge management processes defined by Probst et al. [PRR06].*

Architectural descriptions of both, system and enterprise architectures helped new project members to get an easier overview about the system or company. This helps in terms of »knowledge identification« and »knowledge development« because people are able to find relevant documents by using the architectural descriptions as a guiding map. In the cases of Company C and E this idea of using the architectural descriptions as a knowledge map was pushed even further. They used the architecture models as an interactive information asset base where context related documents were hyperlinked to provide easy and direct access. This alignment of knowledge management to the architecture results in a major improvement of »knowledge distribution«.

7 Concluding remarks

With our research and this paper we develop theses on how architectures influence knowledge management in software projects. According to our literature review it is pretty much the first research project with this specific scope of examination. The conducted number of case studies provides a solid base for our hypotheses but it is still a first exploratory study in that field. As of today, system architectures are mostly used as an instrument to handle the complexity of software systems and enterprise architectures are used as an instrument of governance and business- or IT alignment to safeguard running investments, plan future investments and furthermore to simultaneously reduce costs [LMW05]. In this respect, architectures are already used for knowledge management but in our opinion their potential is not fully tapped on a broad basis.

We can summarize that architectures are a valuable appliance to provide access to explicit knowledge [NT95] of any kind (e.g. information about system requirements, design decisions or algorithms). Our study shows that architectures support a variety of knowledge management processes, if used as knowledge map – especially »knowledge identification« and »knowledge distribution«. They can also vastly improve inner- and inter-team communication by establishing a common vocabulary.

Further investigations will be the definition of blueprints for architectures that are especially suited to support the knowledge management in software projects (or even in general). We also plan to investigate, how specific target audiences can be reached best and how to communicate an architecture to these audiences. In our introduction we also mentioned, that in complex systems, knowledge often needs to be kept alive and accessible for decades. In this regard, historiography is an important aspect. Further research work could deal with strategies how this can be done directly inside architecture models.

8 Acknowledgements

We applied the FLAE norm [THR⁺07] for the sequence of authors of this paper. The credit for a huge amount of the research work goes to Markus May who conducted and evaluated the interviews of the mentioned case studies. Last but not least, we especially thank all of our interview partners for taking time to talk to us.

References

- [AKL99a] F Armour, SH Kaisler, and S Liu. Building an enterprise architecture step by step. *IT Professional*, 1(4):31 – 39, Jul 1999.
- [AKL99b] F Armour, SH Kaisler, and SY Liu. A big-picture look at enterprise architectures. *IT Professional*, 1(1):35–42, 1999.

- [AL01] M Alavi and DE Leidner. Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues. *MIS Quarterly*, 25(1):107–136, 2001.
- [Cou07] North Atlantic Council. NATO Architecture Framework (NAF) Ver 3. 2007.
- [DP98] TH Davenport and L Prusak. Working Knowledge: How organizations manage what they know. *Mcgraw-Hill Professional*, (1), Jan 1998.
- [DW99] W Dröschel and M Wiemers. *Das V-Modell 97: Der Standard für die Entwicklung von IT-Systemen mit Anleitung für den Praxiseinsatz*. Oldenbourg, Nov 1999.
- [Eis89] K Eisenhardt. Building theories from case study research. *Academy of management review*, Jan 1989.
- [Gro09] The Open Group. *Togaf Version 9 - A Manual*. Van Haren Publishing, Jan 2009.
- [Hil00] R Hilliard. IEEE-Std-1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems. *IEEE*, <http://standards.ieee.org>, 2000.
- [Krc90] H Krcmar. Bedeutung und Ziele von Informationssystemarchitekturen. *Wirtschaftsinformatik*, 32(5):395–402, 1990.
- [LMW05] J Lankes, F Matthes, and A Wittenburg. Architekturbeschreibung von Anwendungslandschaften: Softwarekartographie und IEEE Std 1471-2000. *Software Engineering*, pages 43–54, 2005.
- [LS06] S Leuchter and R Schönbein. Die Verwendung von Architectural Frameworks als Vorgehensmodell für die System-of-Systems-Entwicklung. *INFORMATIK 2006. Informatik für Menschen. Beiträge der 36. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*, 2006.
- [May08] M May. Architekturen - Wissensmanagement in IT-Projekten. *Diplomarbeit, Universität der Bundeswehr München*, Jul 2008.
- [MEH01] MW Maier, D Emery, and R Hilliard. Software architecture: Introducing IEEE standard 1471. *Computer*, 34(4):107–109, 2001.
- [NT95] I Nonaka and H Takeuchi. The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation. *Oxford University Press*, Jan 1995.
- [PRR06] G Probst, S Raub, and K Romhardt. Wissen managen: Wie Unternehmen ihre wertvollste Ressource optimal nutzen. *Gabler*, (5. Auflage), Jan 2006.
- [Sch04a] J Schekkerman. How to survive in the jungle of enterprise architecture frameworks: creating or choosing an Enterprise Architecture Framework. *Ebookslib*, Jan 2004.
- [Sch04b] M Schönherr. Enterprise Architecture Frameworks. *Enterprise Application Integration-Serviceorientierung und nachhaltige Architekturen*. Hrsg. S. Aier/M. Schönherr, Berlin: Gito. Reihe: Enterprise Architecture, 2:3–48, 2004.
- [Sch05] J Schekkerman. Trends in Enterprise Architecture 2005: How are Organizations Progressing? *Institute For Enterprise Architecture Developments, Report of the Third Measurement*, Dec 2005.
- [THR⁺07] T Tschardtke, ME Hochberg, TA Rand, VH Resh, and J Krauss. Author sequence and credit for contributions in multiauthored publications. *PLoS Biol*, 5(1):e18, Jan 2007.
- [Zac87] J Zachman. A framework for information systems architecture. *IBM systems journal*, 26(3), Jan 1987.

From Process to Simulation - A Transformation Model Approach

Oliver Kloos, Volker Nissen, Mathias Petsch

Chair of Information Systems in Services
Ilmenau University of Technology
PO Box 10 05 65
98684 Ilmenau, Germany
{oliver.kloos,volker.nissen,mathias.petsch}@tu-ilmenau.de

Abstract: Process models are frequently used to visualise, analyse and control the flow of work in business processes. In principle, the information contained in process models could be used to simulate and optimise business processes with the help of simulation software. However, business process models are created for other purposes than simulation and, therefore, normally do not contain enough information to directly execute simulation studies using them. A process model first has to be converted to a simulation model which holds different information necessary for the simulation. In this paper, a transformation approach is presented which can be used to convert process models based on the event-driven process chain (EPC) notation to different simulation software systems.

1 Introduction

Business process models serve to document and describe the business activities of an organisation, determine the critical points of business processes and/or analyse the requirements for the introduction or implementation of new information systems [Sc00, FL03]. At the same time business process models can be used for other purposes. But, the modelling of business processes is often associated with the rationalisation of these processes (e.g. reducing processing time, lowering process costs etc.) [BRU00, BS04]. Simulation allows us to observe and analyse the behaviour of processes. Therefore, simulation can be used to create and to check the consequences of changes in the processes, for example a different amount of available resources. [BF87, DE00].

In practice, the first step in improving current business processes is the documentation and (usually manual) analysis of existing current processes. The use of these process models for simulation and the optimisation using simulation software are seldom considered while modelling the business processes. Normally the methodologies for modelling processes do not include enough (quantitative) information to model and run simulation studies [St06]. Nevertheless, process models can form a good starting point to create simulation models. The main problem is how a process model can be conveniently transformed to create the foundation of a simulation model. For the purpose of simulation, parts of the pro-

cess model could be aggregated and other parts ignored. On the other hand, certain additional information, such as resources and capacities must be added, which are not available from the original process model but are, nevertheless, important to run a simulation study [Al07]. Our objective is the creation of a transformation model which can be used to normalise the process model and to gather the information necessary for the simulation model. The concept of the transformation model is that of a middleware. Therefore, a process model created without considering the requirements for simulation will be converted to the transformation model, normalised and prepared for simulation and then be transferred to a simulation system. The transformation model reduces the number of transformations from different process modelling notations to various simulation systems. For example if there are four process modelling notations and three simulation systems, there would be twelve transformations. With the transformation model there would be only four transformations to the transformation model and three to the simulation systems. If a direct transformation is used, every process modelling notation needs methods to prepare a process model for a simulation. This preparation is only necessary for the transformation model. Furthermore, the meta-models of the process modelling notation do not include information necessary for simulations. Another problem are resources, for example rooms, which are a simulation constraint, but cannot be added to an extended EPC. So every process modelling notation would need to be expanded if a transformation model is not used. Some process models can of course be directly simulated, for example with the ARIS Toolset, but some process models require additional constraints which cannot be added in process models, so they have to be transformed into simulation models and simulated with other simulation systems. We do not use an existing notation for the transformation model such as the EPC or Business Process Modeling Notation (BPMN). While the EPC is restricted to a function and event sequence, BPMN offers a wide spectrum of methods to model a business process. On the one hand the allowable elements would have to be restricted; on the other hand new artefacts would need to be included. However important information for the simulation would need be added as attributes of the elements which are not visible in the graphical representation of the process model. We introduced a new notation that contains all necessary elements to create a simulation model and that shows all relevant information using a graphical representation to support the human part of the normalisation and preparation of the transformation model.

The remainder of this paper is structured as follows. First, work related to our own research is discussed. Then, we define business process models and simulations and explain the foundations of our transformation model. After that, the notation and different views necessary to generate a valid transformation model are highlighted. Next, the transformation process from a business process model to a simulation model is explained. Finally, our results are summarised and future work is indicated.

2 Related Work

The transformation of business process models to simulation environments has been an active research topic for several years [Gr03, AJ05, HR06, Di07, DD08]. The main problem

concerns the different degree of model detailing [NRS03, HR06]. The following problems especially occur during the transformation from process to simulation models:

- Process models show a higher level of detail in the number and description of process steps, which are seldom necessary for simulation models (process steps are often summarised by one module in simulation models).
- Simulation models require detailed and operationalised data (such as distribution of process time, probabilities, frequency of process steps etc.).

There are several papers that deal with these problems. Greasley [Gr03] used a process map of a prisoner custody process as a conceptual model for the simulation model. First, the process map was created and data were collected such as arrival time and function duration. Next, the process map was transferred to a simulation system. Damij and Damij [DD08] used an activity table of a clinical business process, which represents a table with organisational units as columns and activities as rows. The activities are associated with an organisational unit through the cells and connected with links to create a flow. To create a simulation environment the process was transformed into a flow chart. However, the flow chart presented contains the same information as the activity table but uses a different graphical representation. An and Jeng [AJ05] presented a flow chart, of a supply chain management domain. The functional structure of the flow chart was used to create a simulation model by adding input and output data and their corresponding data repositories. However, the authors also created a system dynamic model, which represents the positive and negative influences of the elements on each other. Both models were used to simulate the supply chain. Dickmann et al. [Di07] presented an approach which supports the process improvements projects. They also used conceptual process models, which were not created for the purpose of simulation but to create simulation models. Therefore, they used questionnaires to gather the information necessary for the simulation model.

All approaches presented use existing process models as a foundation to create simulation models. Since everyone had different notations for the process models, each required its own transformation methods and rules for targeting a simulation system. None of these approaches uses methodology or notation created with the purpose of preparing a process model for simulation. Heavey and Ryan [HR06] analysed the support of simulation in process modelling tools and methods. They outlined that none of them supports the requirements-gathering phase of simulation. As a conclusion they created a process modelling method called Simulation Activity Diagrams, which support the conceptual modelling phase of a simulation project. However this approach does not use existing process models to prepare them for a simulation.

Other papers relevant in the context of the transformation and preparation of a process model for simulation deal with the reduction of process models. While these papers do not focus on simulation they nevertheless can be used for it. Polyvyanyy et al. [RSW08] describe a complexity reduction approach of large EPCs through joining of loops, sequences, and blocks. The approach is based on pattern identification of AND, OR and XOR connectors and the aggregation of functions and events to reduced functions. A similar idea is put forward by Sadiq and Orłowska [SO99]. The objective of their approach is to discover incorrect graphs within process models. The algorithm identifies structurally correct

workflow graphs with the help of so called reduction rules and reduces them to an empty graph. On the other hand Allweyer [Al07] presented a procedure to create a more detailed process model from an imprecise process model. To execute the transformation rules a template for the detailed process model is necessary. The transformation rules use data associated with the imprecise process model to create a detailed process model based on the template. These papers are relevant for the normalisation of the transformation model.

3 Foundations of the transformation model

We define a business process, according to Rosemann, as an enclosed structured, logical series of functions for the handling of a relevant business object. Therefore, a business process model is a model of a structured, logical series of functions executed on an object [Ro96]. Our second object of interest is simulation. Simulation is defined as a method to imitate the operation of a real-world system. Simulation involves the construction of a simulation model and data acquisition, the execution of a simulation study and the analysis of the simulation results [VDI95, BCN05].

The transformation model was derived under consideration of the process modelling using EPC and BPMN notations and three simulation software systems. The core of the transformation model, the activity, is based on three premises. The first premise states that every work in the real world can be described using a combination of an action, an object and one or more resources. An example for such an activity is, 'The salesperson creates the sales order in an ERP system'. The action represents a work step, in this example 'create'. Other examples are 'check', 'order' and 'authorise'. The object is the item on which the action is executed, in the example 'sales order'. Resources are a tool to support the execution of the action, 'salesperson' and 'ERP system' in the example. The combination of these three elements represents the activity in the transformation model. A more complex activity for instance is 'The nurse checks the blood pressure of the patient'. The nurse is the resource and the patient the object. The action however is more than a single verb, in this case 'check blood pressure'. So an action can be a single verb or a combination of a verb and a noun. In contrast to van der Aalst [Aa98] we define an activity differently. We also use a resource to specify an activity, but we divide the task into an action and an object. The case van der Aalst uses would be an instance of an object in a simulation. If the instance is added it would be an activity that is executed.

The second premise states that only a combination of an action, an object and at least one resource can have a processing time. A combination of only two of the three elements cannot have a processing time. A resource is needed to execute an action and the execution must be on an object. Only then the activity being processed can contain a processing time. The third premise states that when there are more actions to be executed than there are resources, a queue will be created. To specify a queue, two pieces of information are needed: the capacity of the queue and the queue processing technique. The activity assumes that if there is no information about the queue, the capacity is infinite and the queue processing technique is 'First In, First Out'. If an activity transforms several objects into a new object, the activity can be extended by input and output information. This

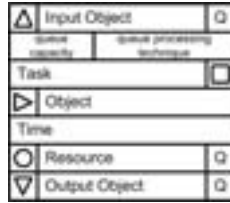


Figure 1: Complex activity

type of complex activity is shown in Figure 1; the 'Q' indicates the quantity needed. The current notation is shown in table 1 and 2.

As shown in the next chapter there is a need to bind resources for more than one activity. For example, assuming a flow chart shows two activities that can only be executed together. Activity one uses resources *A* and *B* and activity two uses resource *A*. Resource *B* can be released after the execution of activity one but resource *A* can only be released after activity two has been executed. Therefore, resource *A* is bound before activity one and released after activity two in the flow chart. An example process that needs a resource bind and release is shown in figure 3.

The last main group of elements in our transformation model are pathways. In the context of a simulation, there are three different types of pathways: parallel, inclusive and exclusive. Every pathway also has gateways containing the condition for the execution of a flow path, except for the parallel pathway. A parallel pathway indicates that two activities can be executed at the same time. The inclusive and exclusive pathways have two variants: probability- and condition-based. One or more flow paths can be executed in the inclusive pathway and only one in the exclusive pathway, according to the occurrence of the gateway. The condition-based gateway checks an attribute of the object. Since the gateway contains the condition, one or more flow paths can be executed. The attribute-based gateway contains the attribute of an object to be checked and the gateway contains a value. If the attribute of an object instance matches the value in the gateway, the flow path is executed. Therefore, only one flow path can be executed. The splitting pathway is displayed with a single border, while the joining pathway has a double border.

To use a condition-based or attribute-based pathway, an attribute of an object needs a value. The value is set by the attribute set. The condition-based pathway can be used to add complex conditions for the execution of flow paths. The attribute-based pathway can be used instead of an exclusive pathway. For example a process model contains two exclusive pathways that use the same condition to determine the flow path. When the first flow path is selected in the first exclusive pathway, then the first flow path of the second pathway must be selected. A process model in the notation of the EPC would use two exclusive operators with the same events, so that the corresponding flow paths are selected. However, the exclusive pathway is probability-based. To select the first flow path in both pathways the second pathway has to be an attribute-based pathway and every flow path of the first pathway needs an attribute set.

Rittgen showed that an OR join in EPC notation has three interpretations [Ri99]. Waiting




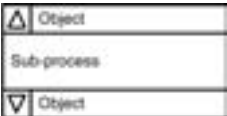
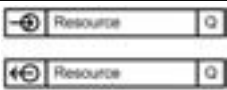





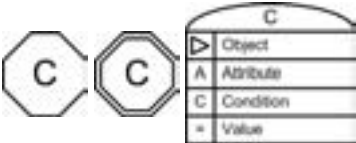

| Notation | Explanation |
|---|---|
|  | The <i>Activity</i> shows the mandatory fields that are necessary for the flow chart of the transformation model. |
|  | The <i>Source</i> creates new instances of the object in the listed time interval and quantity. |
|  | If an instance of an object arrives at the <i>Sink</i> it is removed from the model. |
|  | A <i>Sub-process</i> passes an object to another flow chart. After the execution of the other flow chart, an object is passed back to the flow path in that it is integrated. |
|  | The <i>Resource binding</i> and <i>Resource release</i> is used to bind a resource to the flow path it is integrated. Bound resources can only be used in that flow path. |
|  | The <i>Attribute Set</i> sets a value of an attribute. If an attribute is set the attribute-based and condition-based gateway can check this attribute. |
|  | The <i>Delay</i> delays the execution of a flow path without the need of resources. |
|  | All flow paths between the <i>Parallel Pathway</i> are triggered. Activities in the different flow paths can be executed at the same time. |
|  | One or more flow paths can be triggered in the <i>Inclusive Pathway</i> , depending on the probability of the gateway. |
|  | Only one flow path can be triggered in the <i>Exclusive Pathway</i> . The sum the probability of all gateways of one exclusive pathway must be 100%. |
|  | A flow path is triggered in the <i>Condition-based Pathway</i> , if the condition of the gateway is triggered, so one or more flow paths can be executed. |
|  | The <i>Attribute-based Pathway</i> contains an attribute of an object. If the value in the gateway matches the attribute of the instance only that pathway is executed. |

Table 1: Notation of the transformation model flow chart







| Notation | Explanation |
|---|--|
|   | The successive element of the joining pathway is executed, when all triggered flow paths are completed. ('wait for all') |
|   | The first flow path that is completely executed will be passed to the successive element of the joining pathway. All other flow paths end at the joining pathway. ('first come') |
|   | Every flowpath that is completely executed will be passed to the successive element of the joining pathway. ('everytime') |

Table 2: Joining pathways of the inclusive and condition-based pathway

for the completion of all activated flow paths is the default semantic ('wait-for-all'). Another interpretation is to only wait for the first flow path to be completed. All other flow paths will be ignored ('first-come'). The third interpretation is called 'every-time'. Each completed flow path will be passed to the next element. To cover these interpretations, the joining pathways of the inclusive and condition-based pathway must be specified during the normalisation of the transformation model. Table 2 shows the notation of the joining pathways, which are only available in a consistent transformation model. While the wait-for-all interpretation would be the default, a simulation with the ARIS Toolset shows a different interpretation. If there is an OR split and a corresponding OR join, the OR join passes on more instances than the OR split received, if the sum of the probability of all flow paths is greater than hundred percent. So an OR split in the simulation of the ARIS Toolset is interpreted as every-time.

In addition to the flow chart we define three supplemental views for the transformation model. These views are derived from the first premise of the activity. Therefore, there are an object view, a resource view and an optional action view. The object view only is necessary if a condition-based or attribute-based gateway is used. However, the resource view is inevitable because it contains the quantity of the available resources. All three views are organised in a tree with parent-child relationships. While the action view only organises actions as a function tree; the object and resource view provide additional information. The object and resource view are explained below.

An activity can only be executed with an instance of the object it is composed of. The instance of an object is created by the source. But there could be a process model with an activity that uses an object not created by a source and only used by a single activity. An example for this is a process model that characterises the processing of a sales order. To check if the customer is available in IT system there could be an activity 'check customer number' executed by a salesperson. A source would create an instance of a sales order, but the object of the activity is 'customer number'. To solve this problem, objects are divided into 'process objects' and 'non-process objects' in the object view and flow chart. Process objects are created by a source or the output of an activity and removed by a sink or as used as an input of an activity. However, a non-process object is not created or used as an input and only used in a single activity. So only process objects can be used for

condition- or attribute-based gateways. Therefore, the process objects have to contain the attributes in the object view, so that they can be used in the flow chart. Because non-process objects cannot be used by gateways there is no need to add attributes to these objects. To symbolise that a non-process object is used in the flow chart, the triangle identifying the process object is mirrored.

Resources can be divided into four different types: human, stationary, movable and immaterial resources. This division results in three criteria human vs. non-human, material vs. immaterial and stationary vs. movable. While a human resource is always a human, material and movable, whether it moves or not, an immaterial resource is immaterial and non-human. Because of the immateriality there is no place in the real world it is situated, ignoring the fact that it could be saved in a data storage. The stationary and moveable resources are non-human and material, whereas the first is stationary, the second is moveable. An example of a human resource is a doctor or salesperson. A stationary resource could be a room or a machine, while a movable resource is for example a forklift truck. An example for the immaterial resource is an IT system.

All four types are children of the type resource, for example a room would be the child of the stationary resource. To execute a simulation study, it is necessary to define the quantity of each resource used. An unlimited resource does not need to be included in a simulation study. While the flow chart only holds the number of necessary resources for each activity, the resource view contains their quantities. Currently in development is the attribute definition for the four types of resources, which can be transformed for a simulation software system. Humans could have a shift schedule, while stationary resources could have an availability or capacity. Stationary resources could also be used to create a layout. Using this layout the human and movable resources could be assigned paths they must take to get to another stationary resource.

4 Transformation Process

The first step before executing a simulation study is the definition of the problem, representing the purpose of the simulation. Based on the defined problem an individual or multiple process models are selected for analysis in the simulation study. It should be noted here that the use of the transformation model requires existing process models as input. The next step is an automatic transformation of the process models to the notation of the transformation model. Corresponding transformation rules are mentioned below. Whereas the process models are created for other purposes, the transformation model has to be fed with additional information necessary for the execution of a simulation study. Therefore, the transformation model created via the transformation rules is only a conceptual model. The conceptual model is characterised that it uses the notation of the transformation model, but the elements used lack information, such as execution time in the activity or the probability in the exclusive gateway, which were not found in the process models. Therefore, data has to be collected and used to normalise the transformation model into a consistent model. On the one hand all elements used in the consistent transformation model have the necessary information, but on the other hand normalisation rules are executed such that elements

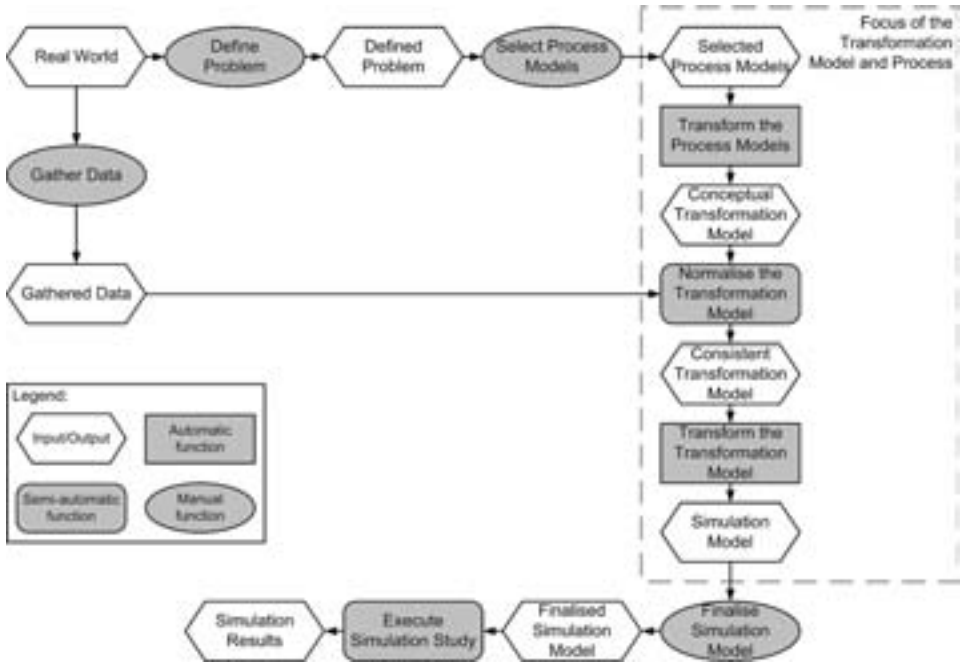


Figure 2: Procedure that leads from process models to simulation

could be changed or added to the consistent transformation model. Examples for the normalisation are shown below. This normalisation step can be executed semi-automatically. While the merge of activities can be executed semi-automatically, additional resources for the activities have to be added manually. When a consistent transformation model has been created, the model can be automatically transformed into a simulation model. This simulation model depends on the software system being used to execute a simulation study. Before the simulation study can be executed, the simulation model can be expanded with additional information, depending on the simulation software system. Examples for this information are the duration of the simulation, resource utilisation or diagrams to visualise the simulation results. With the finalised simulation model the simulation study can be executed to gather knowledge, which can in turn be used to interpret the real world.

The procedure of the described transformation process is shown in figure 2. The figure also shows the main focus of the transformation model, the transformation from process models, the normalisation of the transformation model and the transformation to a simulation model. While both transformations can be executed automatically, normalisation of the transformation model needs human interactions and is therefore only semi-automatic. With this procedure there is no need to change the existing process models, but only to adapt and enrich their contents in the process of converting them to a simulation model.

Next, examples are given for using the transformation model and process. The examples cover transformation rules from the EPC and rules to normalise the transformation model.

While transformation rules for BPMN are currently created, we identified 23 transformation rules to convert the elements of an EPC to the elements of a flow chart of the transformation model. The transformation rules for a specific simulation software system are not discussed in this paper.

Normally, resources used in the EPC cover humans and information systems. Stationary or movable resources are therefore initially unavailable. Focussing on information systems first, the information system in the EPC can be transferred to the resource view of the transformation model as a child of the immaterial resource. The quantity of the information system is one. To avoid the fact that only one operation can be executed at the same time with the information system, the attribute capacity has to be assigned. However, if the focus of the simulation study is not on the utilisation of information systems, it can be more appropriate not to transfer the resource information system from the EPC to the transformation model at all, or remove the resources during normalisation. Therefore, some analysis is required during this step to decide on the better transformation option.

As for humans in the process, it is suggested to use organisational units from an organigram, which executes the functions of the EPC. The two alternatives discussed here are positions and person types. While positions cover a person of a specific department such as a salesperson, purchaser or accounting clerk, person types deal with hierarchy, such as supervisor, staff member or proposer. A specific person can, therefore, have a particular person type and, simultaneously, a certain position in the company. To prevent the situation in which there are more humans represented in the simulation model than in the actual enterprise, only one of these alternatives should be implemented in the transformation model. The transformation rule for both possibilities is the same. A position or person type is added as a child of the human resource in the resource view. The quantity of humans in the resource view can be determined by summing up the amount of people assigned to the position or person types associated with the resource view.

All start events are converted to sources and all end events are converted to sinks. The subject used in the event is the object in the source and sink. Events after a separating OR or XOR operator are used in the gateways of the inclusive and exclusive pathways. All other events are removed and not implemented in the transformation model. The separating operators are converted to the corresponding separating pathways and the joining operators to joining pathways. An AND operator will be a parallel pathway, an OR operator an inclusive pathway and an XOR operator an exclusive pathway. An automatic conversion of operators for condition- or attribute-based pathways is not possible. The activity is created based on the function. Positions, person types or information systems will be the resources of the activity. A resulting transformation rule would be: 'If an XOR-Operator has one successor and more than one predecessor, then it will be an exclusive split pathway.'

An example for the transformation of an extended EPC to the transformation model is shown in figure 3. While the first flow chart shows an EPC, the second flow chart shows a conceptual transformation model, which is automatically created. To execute the three activities a room is needed, so the rooms are added to the third flow chart, the consistent transformation model. The room and the dental assistant are reserved for the flow path. All three activities have to be executed, so a second instance can only use the reserved room after it is released. A semi-automatic transformation rule would be: 'If a resource is

used in multiple activities in a sequence, determine if the resource should be reserved for the flow path.’ If the extended EPC is simulated, there is a logical problem with the room, for example only one room is available and there are two instances. Instance one executes the first and second functions. Then the second instance executes function one and after that the first instance would execute the revision. In reality the first patient would have to leave the room after the examination, so the second patient can have his preliminary enquiry. Then the second patient would leave the room, so that the first patient can have his revision. To prevent this sequence the room is reserved in the consistent transformation model.

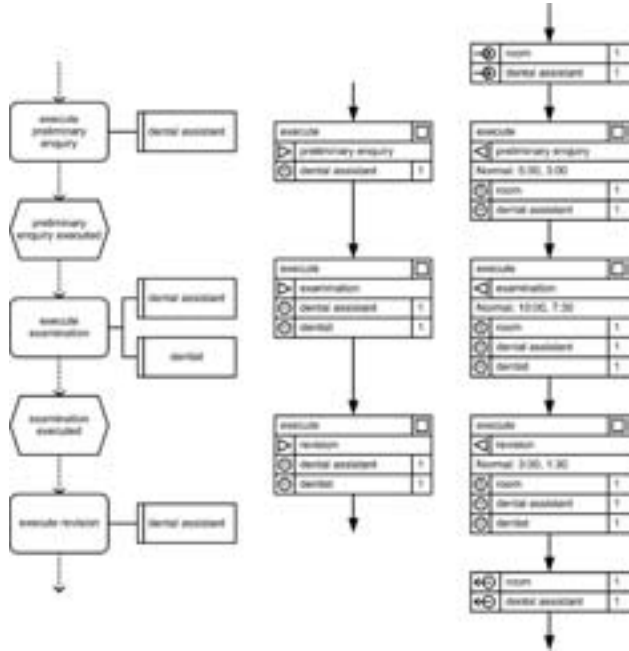


Figure 3: Example process with an extended EPC and the consistent transformation model

Another approach for normalising the transformation model is to add attributes to the objects and change the pathways. An inclusive pathway can be converted to a condition-based pathway and an exclusive pathway can be converted to an attribute-based pathway. This approach can be used when different input streams are the focus of the simulation study. The values of the object attributes depend on the input stream and therefore the probability of the pathway. This approach can help reduce the maintenance effort of the simulation models. There could also be different terms used in the process model, for example bill or invoice. These terms have to be harmonised during the normalisation of the transformation model.

An example for a more complex normalisation is shown in figure 4, based on Rosemann [Ro96]. The EPC is modelled according to the Guidelines of Business Process Modelling [BRU00]. However, the event ‘goods arrived’ has to be a source. But goods only arrive

when an order is created. If this process is simulated goods may arrive without the creation of an order. The normalisation of the conceptual transformation model converts the joining parallel pathway and the start event into a delay. So every order has a delivery time. On the one hand this example shows on a manual normalisation in which resources are such as the loading ramp or the forklift truck are added. On the other hand an automatic normalisation process is shown in which the parallel join pathway and the source are replaced by a delay. The corresponding normalisation rule would be: 'If a parallel join pathway has one activity as a predecessor and one source as predecessor, then it is replaced by a delay.'

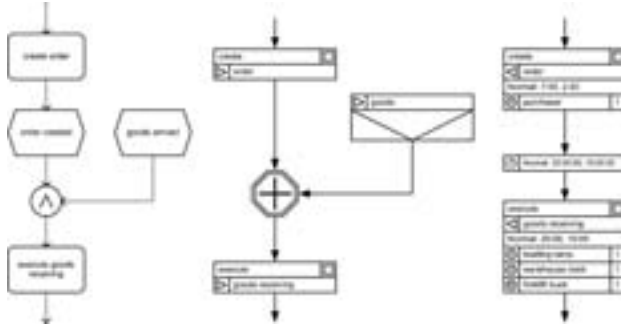


Figure 4: Example for the normalisation of the transformation model

After the normalisation of the transformation model has been executed, a direct conversion to a simulation model becomes possible.

5 Conclusion and Future Work

This paper introduced a process and notation for semi-automatically transforming process models into simulation models where the process models were not originally created with simulation in mind. Process models are converted to transformation models and normalised, in order to ultimately transform them into simulation models using arbitrary simulation software. The transformation model contains a flow chart and three views, which were presented here. Also, some initial transformation rules were demonstrated with reference to process models in the notation of the extended EPC. Additionally, some information was provided on how to reduce the complexity of the original process model and further normalise the transformation model.

The transformation model introduced is still a work in progress and the next step in our research is the creation of a meta-model for the flow chart and the three views in the transformation model. Based on meta-models the transformation rules could be created for the transformation of process models into the transformation model. However, specific linguistic transformation rules are necessary if a meta-model element contains more than one element of the transformation model. These rules use word classes and cases. While these rules are only valid for a single language or perhaps also for a single notation, a

research goal is the creation of linguistic transformation rules for different languages and notations. As for tool support of the notation, we are currently developing a prototypical implementation based on the Eclipse framework. At the moment the prototype supports modelling of the flow chart, an automatic conversion of the EPC markup language [MN05] to the flow chart and automatic export to the simulation software Arena. Preliminary transformation rules for BPMN are identified but not described in concrete transformation rules. In future research, these transformation rules for BPMN and the UML activity diagram will be developed based on the meta-model, so that the most frequently employed modelling notations can be conveniently converted to a simulation model.

Other research themes to be addressed include developing further rules to normalise the transformation model. A starting point here is the creation of a procedure model, which covers all steps of the normalisation. Those steps must include reducing the complexity of the transformation model so that only the relevant aspects for the simulation remain as well as adding relevant additional information through attributes of the resources.

Currently, the transformation rules to create a simulation model take into account process-oriented simulation software. However, in the domain of logistics and production, resource-oriented simulation software is dominant. Therefore, some initial research is currently done to create transformation rules for this and other types of simulation software.

References

- [Aa98] Aalast, W.M.P. van der: The Application of Petri Nets to Workflow Management. In: *Journal of Circuits, Systems and Computers* 8(1), 1998; pp. 21-66.
- [Al07] Allweyer, T.: Erzeugung detaillierter und ausführbarer Geschäftsprozessmodelle durch Modell-zu-Modell-Transformationen. In (Nüttgens, M.; Rump, F. J.; Gadatsch, A. Eds.): *6. GI-Workshop EPK 2007 : Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten*, St. Augustin, 2007; pp. 23–38.
- [AJ05] An, L.; Jeng, J.-J.: On developing system dynamics model for business process simulation. In (Kuhl, M.E.; Steiger, N.M.; Armstrong, F.B.; Joines, J.A. Eds.): *Proceedings of the 2005 Winter Simulation Conference*, Orlando. IEEE Press, Piscataway, 2005; pp. 2068–2077.
- [BCN05] Banks, J.; Carson, J.; Nelson, B.L.: *Discrete-event system simulation*. Prentice Hall, Upper Saddle River, 2005.
- [BF87] Bratley P.; Fox, B.L.: *A guide to simulation*. Springer, New York, 1987.
- [BRU00] Becker, J.; Rosemann, M.; Uthmann, C.: Guidelines of Business Process Modeling. In (Aalst, W.; Desel, J.; Oberweis, A. Eds.): *Business Process Management : Models, Techniques, and Empirical Studies*. Springer, Berlin, 2000; pp. 241–262.
- [BS04] Becker, J.; Schütte, R.: *Handelsinformationssysteme : Domänenorientierte Einführung in die Wirtschaftsinformatik*. Redline Wirtschaft, Frankfurt, 2004.
- [DD08] Damij, N.; Damij, T.: Improvement of a clinical business process. In (Al-Dabass, D.; Nagar, A.; Tawfik, H.; Abraham, A.; Zobel, R. Eds.): *Second UKSIM European Symposium on Computer Modeling and Simulation*, 2008. EMS '08, Liverpool. IEEE Press, Piscataway, 2008; pp. 305–310.

- [DE00] Desel, J.; Erwin, T.: Modeling, Simulation and Analysis of Business Processes. In (Aalst, W.; Desel, J.; Oberweis, A. Eds.): *Business Process Management : Models, Techniques, and Empirical Studies*. Springer, Berlin, 2000; pp. 247–288.
- [Di07] Dickmann, C.; Klein, H.; Birkhölzer, T.; Fietz, W.; Vaupel, J.; Meyer, L.: Deriving a Valid Process Simulation from Real World Experiences. In (Wang, Q.; Pfahl, D.; Raffo, D.M. Eds.): *International Conference on Software Process, ICSP 2007, Minneapolis*. Springer, Berlin, 2007, pp. 272–282.
- [FL03] Frank, U.; van Laak, B.L.: Anforderungen an Sprachen zur Modellierung von Geschäftsprozessen. *Arbeitsberichte des Instituts für Wirtschaftsinformatik Nr. 34*, Universität Koblenz Landau, 2003.
- [Gr03] Greasley, A.: Using business-process simulation within a business-process reengineering approach. In: *Business Process Management Journal* 9(4), 2003; pp. 408–420.
- [HR06] Heavey, C.; Ryan, J.: Process modelling support for the conceptual modelling phase of a simulation project. In (Perrone, L.F.; Wieland, F.P.; Liu, J.; Lawson, B.G.; Nicol, D.M.; Fujimoto, R.M. Eds.): *Proceedings of the 2006 Winter Simulation Conference, Monterey*. IEEE Press, Piscataway, 2006; pp. 801–808.
- [MN05] Mendling, J. Nüttgens, M.: EPC Markup Language (EPML) : An XML-Based Interchange Format for Event-Driven Process Chains. In: *Information Systems and E-Business Management* 4(3), 2005; pp. 245–263.
- [NRS03] Neumann, S.; Rosemann, M.; Schwegmann, A.: Simulation von Geschäftsprozessen. In (Becker, J.; Kugeler, M.; Rosemann, M. Eds.): *Prozessmanagement - Ein Leitfaden zur prozessorientierten Organisationsgestaltung*. Springer, Berlin 2003; pp. 449–468.
- [Ri99] Rittgen, P.: Modified EPCs and their semantics. *Arbeitsberichte des Instituts für Wirtschaftsinformatik Nr. 19*, Universität Koblenz-Landau, 1999.
- [Ro96] Rosemann, M.: Komplexitätsmanagement in Prozeßmodellen : Methodische Gestaltungsempfehlungen für die Informationsmodellierung. Gabler, Wiesbaden, 1996.
- [RSW08] Polyvyanyy, A.; Smirnov, S.; Weske, M.: Reducing the complexity of large EPCs. In (Loos, P.; Nüttgens M.; Turowski, K.; Werth, D. Eds.): *Modellierung betrieblicher Informationssysteme : Modellierung zwischen SOA und Compliance Management*. Köllen, Bonn 2008; pp. 195–207.
- [Sc00] Scheer, A.-W.: *ARIS - Business Process Modelling*. Springer, Berlin, 2000.
- [SO99] Sadiq, W.; Orłowska, M.E.: Applying Graph Reduction Techniques for Identifying Structural Conflicts in Process Models. In (Jarke, M.; Oberweis, A. Eds.): *Advanced Information Systems Engineering*. Springer, Berlin 1999; pp. 195–209.
- [St06] Staud, J.: *Geschäftsprozessanalyse : Ereignisgesteuerte Prozessketten und objektorientierte Geschäftsprozessmodellierung für Betriebliche Standardsoftware*. Springer, Berlin, 2006.
- [VDI95] VDI: *Simulation von Logistik-, Materialfluß- und Produktionssysteme - Begriffsdefinitionen*. Verein Deutscher Ingenieure, Berlin, 1995.

Controlled Flexibility and Lifecycle Management of Business Processes through Extensibility

Sören Balko¹, Arthur H.M. ter Hofstede², Alistair Barros³, Marcello La Rosa², and Michael Adams²

¹ SAP AG, Walldorf, Germany

Soeren.Balko@sap.com

² Queensland University of Technology, Brisbane, Australia

{a.terhofstede,m.larosa,mj.adams}@qut.edu.au

³ SAP Research, Brisbane, Australia

Alistair.Barros@sap.com

Abstract. Vendors provide reference process models as consolidated, off-the-shelf solutions to capture best practices in a given industry domain. Customers can then adapt these models to suit their specific requirements. Traditional process flexibility approaches facilitate this operation, but do not fully address it as they do not sufficiently take controlled change guided by vendors' reference models into account. This tension between the customer's freedom of adapting reference models, and the ability to incorporate with relatively low effort vendor-initiated reference model changes, thus needs to be carefully balanced. This paper introduces *process extensibility* as a new paradigm for customizing reference processes and managing their evolution over time. Process extensibility mandates a clear recognition of the different responsibilities and interests of reference model vendors and consumers, and is concerned with keeping the effort of customer-side reference model adaptations low while allowing sufficient room for model change.

1 Introduction

In many industries, a company's environment, such as customer demand, technological innovations and regulatory conditions tend to change frequently and sometimes rapidly. By being able to flexibly adapt their processes to changes, agile businesses set themselves apart from their competitors. Naturally, Business process management suites (BPMS) offerings need to facilitate flexibility at low costs. At the same time, companies still wish to benefit from standardized best practices, represented through vendor-provided reference processes. The business process community has come up with numerous flexibility techniques to incorporate change into business processes, e.g. [RA07,GAJVL08,RRKD05,AWG05,AHEA06,EKR95]. These approaches cover both design time and runtime changes and provide formal frameworks for how to constrain changes. However, many established process flexibility approaches suffer from shortcomings with respect to process lifecycle management in general, and to the costs associated with changing business processes, specifically. Some techniques propose that BPMS customers alter reference processes "in place" in order to customize them to their

needs (*patching* use-case) [FLZ06]. Others suggest to use reference processes merely as templates for developing company-specific processes (*blueprinting* use-case) [SN00].

Neither of these approaches can realistically succeed in large-scale software roll-outs, involving hundreds of reference processes with an even higher number of customer adaptations on top. This is because making changes to reference processes goes along with substantial costs for carrying out these changes and later maintaining the resulting processes. Whenever a BPMS vendor ships a new reference process version to incorporate corrections or to address new requirements, existing customer adaptations have to be re-applied at great cost. Similarly, multiple independently defined adaptations have to be consolidated within a single process, for instance when two customer departments change a cross-departmental reference process independently at different points in time.

This paper introduces the concept of *process extensibility* as a new paradigm for customizing reference processes and managing their evolution over time. Process Extensibility mandates a clear recognition of the different responsibilities and interests of reference model vendors and reference model consumers, and is concerned with keeping the effort of reference model adaptations at the customer side low while allowing sufficient room for model adaptation. BPMS vendors *own* (i.e. define and maintain) reference process models, while BPMS customers own and run extensions thereof. These extensions constitute separate customer-defined “delta improvements” which hook up to a reference process through late binding mechanisms. When adhering to some plain compatibility rules, both reference processes and extensions can be patched (i.e. maintained) by their respective owners without ever having to be “re-wired”. The vendor remains in the “driver seat” to update reference content, letting customers easily benefit from state-of-the-art best practices. By automatically applying existing customer extensions to patched referenced processes, the cost of rolling out new BPMS releases is greatly reduced.

The rest of this paper is organized as follows. Section 2 outlines the extensibility approach and its benefits over existing flexibility approaches. Next, Section 3 provides a taxonomy to classify flexibility approaches. Section 4 identifies extensibility patterns that occur along a process’ control flow and data perspective, while Section 5 sets an agenda for future research in this area. The paper concludes with a section on related work and a summary.

2 Extensibility

The general concept of making processes more flexible by allowing deviation from their hard-wired business semantics has been around for some time. Requirements like customization, exception handling, re-use, etc. have led to different technological approaches, namely *From-Scratch Design*, *Patching*, *Blueprinting*, *Ad-Hoc Changing*, and *Runtime Settings*.

2.1 Proposal

Extensibility is a new approach to support process flexibility which specifically addresses customization of reference content. Unlike existing approaches, extensibility

clearly designates responsibilities for the process and extensions thereof. Reference processes may be patched (bugfixed, updated) by the vendor only. Customers receive reference processes as read-only shipped content which is only updated as part of a software release.

Customers then customize reference processes to their needs by independently defining and deploying extensions which solely constitute “deltas” (process fragments). Extensions exist alongside the (reference) processes. At runtime, an extensibility framework dynamically invokes the defined extension(s) for a process. Multiple extensions to a single process can be independently defined (e.g. by different customer departments) to be deployed in isolation (i.e. at different points in time). As the extensibility framework automatically controls the interplay between multiple different extensions and their target (reference) process, there is no need to statically integrate all extensions upfront.

Both reference processes and extensions can be “patched”, thereby spawning new versions. Patches to a reference process should adhere to some compatibility rules that allow the new version to support, wherever possible, existing customer extensions. The extensibility framework takes care of automatically incorporating all customer extensions that were defined atop any old version of the patched reference process. Similarly, extensions need to follow some compatibility rules. These rules constrain to what extent the business semantics of a reference process can be deviated from. Apart from “safe” implicit compatibility rules, the BPMS vendor may define more relaxed explicit constraints.

Figure 1 illustrates a vendor-shipped reference process (left) that is customized with extensions of the customer’s HR and Sales departments. The initial reference process is a sequence of three activities: “HR Task”, “Sales Task”, and “ERP Service”. The first extension replaces “HR Task” with a subflow comprising the existing “HR Task” followed by some organizational chart lookup (“OrgChart Service”).

As part of a new release, the vendor ships a patched reference process that conditionally performs an automated “CRM Service” instead of the (manual) “Sales Task”. The patched reference process is compatible with any extensions defined on its predecessor version. The extension framework needs to automatically route the patched reference process to existing extensions, where applicable (here “HR Task” → “HR Extension”). Independently to the vendor shipment, the customer may have replaced the manual “Sales Task” with an automated “Sales Service”. The earlier defined “HR Extension” was also refined to introduce a four-eyes principle. That said, patches may be applied to both the original reference process and a customer extension.

Extensibility is a prerequisite for proper process lifecycle management where the reference content vendor and the customer represent distinct parties having different requirements and obligations:

Vendor The vendor is responsible for (1) delivering correct reference processes (“shipped content”) that represent generalized best practices. He also needs to (2) maintain that content, i.e. ship patches when bugs are detected or requirements change. Finally, (3) the vendor should provide the means to have its content “customized” to a customer’s needs. From his perspective, it is vital to ensure that reference process change is controlled.

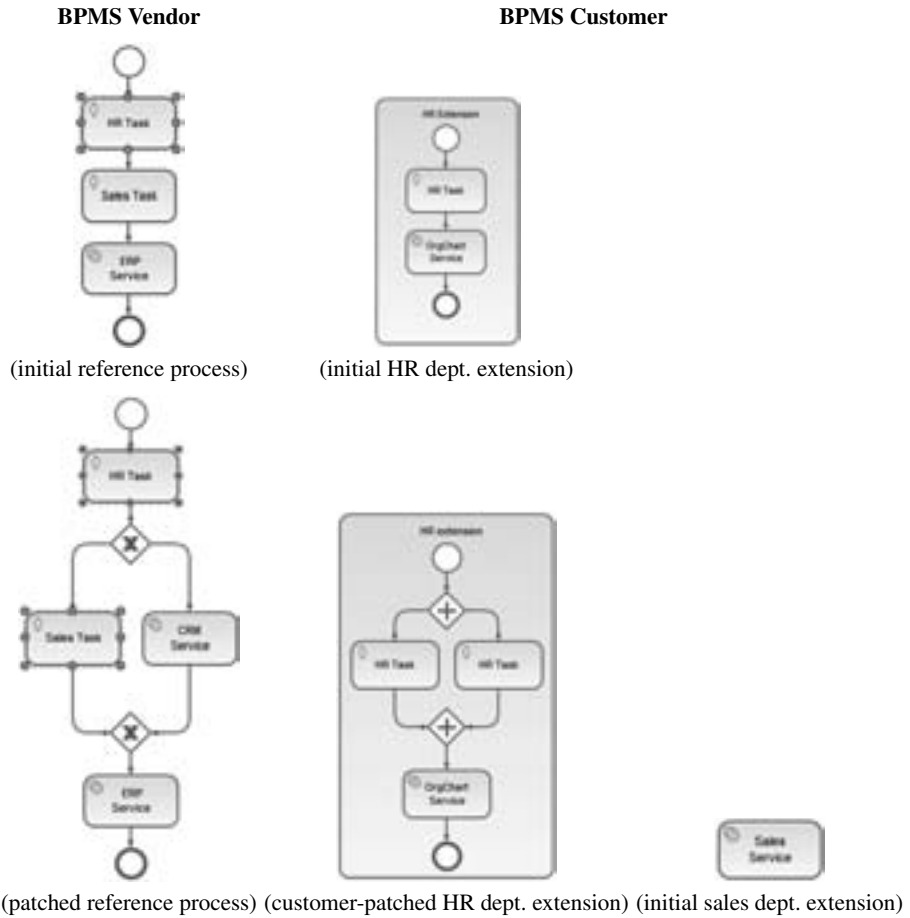


Fig. 1. Extensibility Example

Customer Customers engage their IT team to customize a BPMS release (they may also hire contractors to do so). In regard to reference processes, that includes (1) changing settings which deviate in the customer's landscape, (2) reducing complexity by removing functionality that is not needed, and (3) adding new functionality for requirements which are not yet covered.

End users essentially run processes (i.e. start new instances or are involved in process activities). There are often multiple end user roles that (1) interact with the same process but (2) have their distinct customization requirements. For instance, a legal department could ask for fine-grained logging within audit-sensitive processes, whereas the IT department may be interested in getting notified of technical process failures.

Extensibility offers *controlled flexibility* for the different parties that design, customize, and run processes and is motivated by the specific concerns these parties typically have.

For instance, the vendor must be able to easily patch shipped content without introducing extra, per-customer development costs or significantly increasing the cost of ownership at the customer. Last but not least, the vendor will want to disallow arbitrary changes to this content to avoid mistakes on the customer that which are very difficult to support. In turn, customers are essentially concerned with running their businesses while keeping IT costs down. While flexibility does have its merits, customers also want to build their business on best practices. Besides, customers have a vital interest in correct, law-conforming processes where customizations are guaranteed not to distort the basic functionality.

2.2 Benefits

Technically, the vendor ships reference processes that incorporate “extension points” which are pre-planned artifacts where customers can incorporate their extensions. Extension points apply to almost any dimension of modeling business processes, including control flow, data flow, resources, rules, security, etc. We will give extensibility examples for some perspectives below. Extensibility comes with a number of significant advantages over existing flexibility approaches, notably:

Extensibility (1) offers a lifecycle model for controlled flexibility taking into account obligations and concerns of different parties involved in designing, customizing and using business processes. It helps avoiding errors at the customer side and reduces maintenance costs (*Controlled Change*). Customers automatically (2) benefit from best practices within shipped reference processes. In particular, the vendor can set extension points in a way that the basic business objective of the reference process cannot be tampered with by customer-defined extensions (*Best Practices Adoption*). Reference processes may be (3) subject to patching. Extensions defined on an old version of some process transparently apply to any new version. Reference processes can thus be fixed without losing (or having to manually re-apply) their extensions (*Supportability*).

Instead of using reference processes as templates for newly created processes, (4) extensions consume fewer resources at runtime. This is because an extension solely constitutes a small “delta”. As a side effect, this model is ideally suited for process outsourcing where reference processes are remotely run at *SaaS* providers (*Resource Consumption*). Extensibility allows multiple people (at the customer side) to (5) independently define “additive” extensions to the same reference process. This greatly improves separation of concerns between different business departments. As a result, multiple extensions can be independently defined at different points in time (*Multiple Extensions*).

If desired, vendors may (6) ship their processes as “black boxes”, only exposing interfaces and extension points. This may be desirable if details in the reference content constitute significant intellectual property which is not to be disclosed (*Intellectual Property*). Reference processes may also be purely documentary models that are not executed in a proper BPMS runtime but rather as a coded application. The customer (7) may still want to extend these “application processes” with proper process models. With some application instrumentation to add extension points, extensibility may even help in bridging these platform and paradigm differences (*Application Extension*).

Finally, the (8) meta-process of defining extensions is of interest itself, as it reveals how a customer deals with business change. Mining the logs of a meta-process could help the customer optimizing its business by getting answers to questions like: Which line of business is most often subject to change? Which user roles require most change to reference processes? (*Flexibility Mining*)

3 Taxonomy

Common process flexibility approaches can be classified with respect to a number of dimensions, the most important being (1) the *primary use-case* which outlines the main purpose and most frequent usage, (2) the *parties* (vendor, customizer, end user) that are affected, (3) the functional *role* descriptions of each participant, (4) the *lifecycle* stages (design time, runtime) of the process, (5) the *constraints* that restrict what can be done, and (6) the *scope* (process type, instance, version) within which the flexibility technique operates. Existing flexibility techniques can be classified with this taxonomy, which helps in understanding their differences. It also outlines the contribution of extensibility to the overall picture. We specifically discuss the differences between *from-scratch modeling* of new processes, *patching* existing processes, re-using a vendor-provided template to develop a new business process (*blueprinting*), performing *ad-hoc changes* of process instances at runtime, modifying (technical) *runtime settings*, and *extending* reference processes:

From-Scratch Modeling Modeling a business process “from scratch” is typically the result of analyzing and documenting existing processes. Most importantly, there is no pre-existing reference process to build upon. Instead, a new process is modeled and then successively refined, following a top-down approach. Bottom-up approaches start with modeling detailed process fragments which are later aggregated into larger end-to-end business processes.

Strictly speaking, this approach traditionally does not constitute a flexibility use-case. However, modeling a (reference) process from scratch is a prerequisite for any other flexibility technique. It is usually business analysts who start modeling from scratch. Both the vendor and its customers may perform this use-case (for reference processes and customer processes, respectively). Newly modeled processes are not subject to any constraints, except for the inherent restrictions of the chosen modeling standard.

Patching Occasionally, process models that have been previously deployed to a BPMS runtime engine, may need to be altered. The vendor may have to patch reference processes to fix bugs or simply to address new requirements. Customers may want to patch their processes to incorporate various changes in their business. Patching is closely related to versioning where the affected process will be labeled with a new version number.

Both IT (process developer) and business (domain expert) users may want to patch a process. Patching is a design time operation but will only take effect after deploying the patched process version into the BPMS runtime engine. There are some constraints that limit what can be changed when patching a process. Firstly, interface compatibility must be preserved such that client processes do not have to be

adapted to cope with the change. Secondly, existing extension points must be retained in the patched version such that extensions transparently apply to the patch.

Blueprinting Vendor-delivered reference processes often constitute best practices rather than ready-to-run processes. Blueprinting uses reference processes as a “master” for newly modeled processes. Technically, the reference process is physically copied to a blank process model where it is further refined. While being fully flexible in what changes can be done from there on, BPMS vendors will not be able to support those changes. That is, customers will have to manually apply all changes in a new reference process version in their derived processes (copies). Altogether, blueprinting is a design time operation where customers adapt vendor-delivered reference processes to their needs (as opposed to extensibility which relies on late binding mechanisms). Unlike patching, customers perform modifications on physically separate copies of the template and rather create new variations that are independent from (and do not overwrite) the original process.

Ad-hoc Changing Sometimes, end users have to deviate from the behavior of the process instances they are involved in. Actually, human-driven processes often run into exceptional situations. End users then need to (implicitly) alter the process model for their specific instance, thus deviating from its original business semantics.

There are some constraints around ad-hoc changes, mostly affected with role-related restrictions and instance migration issues. That is, ad-hoc changes alter the models of running process instances. Consequently, ad-hoc changes must allow for automatically migrating the instance state to the altered model. Typically ad-hoc changes affect only a single process instance. The altered process model is kept temporarily, i.e. for the life time of that instance.

Runtime Settings Some environmental settings globally hold for all processes and, when changed, need to immediately apply to both all running processes and newly started instances. Those settings include modifications to organizational charts, security settings and other technical configurations. In most cases, these settings are not even part of any process model such that there is essentially no design time aspect here. Those changes are typically done by system administrators.

Extensibility constitutes a separate flexibility approach where customers define process extensions as deltas (process fragments) on top of reference processes. The primary use-case behind extensibility is customization where the customer adapts a given reference process to its business needs. Various customer roles may define process extensions, each with different objectives. Domain experts from specific organizational lines (e.g. Sales, Procurement, Manufacturing, etc.) may independently define extensions to adjust a cross-organizational process to their needs. A customer typically defines extensions in a design time environment, even though that does not rule out the option of having a runtime user interface to let end users specify extensions in an ad-hoc fashion. Extensibility is subject to some constraints, either originating from implicit compatibility rules or explicitly from modelled extension points within reference processes. Table 1 classifies existing flexibility techniques according to the dimensions introduced and positions extensibility as a new approach.

| | Use-Case | Party | Role | Lifecycle | Constraints | Scope |
|------------------------|---|-------------------|-------------------------------------|-----------------------|------------------------------------|-----------------------|
| Designing from Scratch | Business process analysis | Vendor, Customer | Business analyst, Process architect | Design Time | – | new process |
| Patching | Changing requirements, Bugfixing | Vendor, Customer | Process developer, Domain expert | Design Time | Extension/ interface compatibility | new version |
| Blueprinting | Customization, Adoption of best practices | Customer (Vendor) | Process developer, Domain expert | Design Time | – | new process |
| Ad-Hoc Changing | Handling of exceptional cases | Customer | Task owner, Process administrator | Runtime | Instance migration, Role | single instance |
| Runtime Settings | System-wide settings | Customer | System administrator | Runtime | – | all running instances |
| Extensibility | Customization, Adoption of best practices | Customer | Domain expert, IT department | Design Time (Runtime) | Extension point | all future versions |

Table 1. Process Flexibility Taxonomy

4 Extensibility Patterns

Conceptually, extensibility is open to different process perspectives. This section identifies some frequent extensibility patterns in the control flow and data flow perspectives. Without loss of generality, we use a BPMN-like notation to illustrate these use-cases.

4.1 Control Flow Perspective

Many extensibility use-cases do in some way alter the control flow by adding or replacing process fragments by customer extensions. Extensions may also skip or even re-arrange existing reference process branches. Multiple variants exist, most notably for how to spawn (conditionally, (a)synchronously, etc.) and merge back extension flow (with or without synchronization).

In this paper, we solely consider *Usage Extensibility* which is the most straightforward way of creating control flow extensions. Usage extensibility applies to activities, denoting atomic tasks (either performed automatically or by a human actor) or referencing nested subflows. The idea is to have an extension *replacing* an activity A of the reference flow by another activity A' . Technically, the to-be-replaced and replacing activities A and A' need to expose compatible interfaces (for the data flow) to have the extensibility framework seamlessly perform the replacement without human intervention at runtime.

Figure 2 depicts a “Make to Order” reference process derived from a public SAP *Solution Composer*⁴ business scenario map. Make to Order specifies a vendor-side pro-

⁴ <http://www.sap.com/solutions/businessmaps/composer/index.epx>

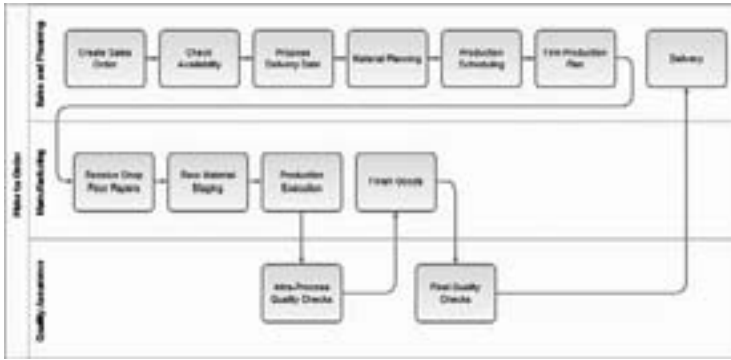


Fig. 2. “Make to Order” Reference Process

cess in discrete industries where a good is manufactured upon an incoming order from a customer. On the vendor side, activities are performed by three different roles: (1) sales department, (2) manufacturing, and (3) quality assurance. After negotiating delivery dates and completing the production planning, manufacturing ultimately produces the good with interleaved quality checks for the production process and final checks for the good itself. At customization, this process is extended to optionally modify those quality gates depending on the order volume. That is, for high-volume orders a four-eyes quality check applies as part of the final checks. For this purpose, the extension replaces the “Final Quality Checks” task by the subflow depicted in Figure 3 (left).

Usage Extensibility captures a wide range of customization use-cases and can be applied in a straightforward way. In fact, by substituting atomic activities through subflows, it allows the incorporation of structurally complex customer extensions into reference flows.

4.2 Data Flow Perspective

Unlike control flow, data flow is implicitly incorporated into process models. It affects the process’ data context, activity interfaces, data mappings, decision gateways, and message correlations. One frequently observed requirement revolves around *Field Extensibility* which deals with (compatibly) complementing data interfaces both from a service provisioning and consumption perspective. That is, customers may wish to customize the reference process in a way that it receives (passes on) additional parameters from inbound (outbound) messages (services). New clients may interact with the process through the field-extended interface. In turn, compatibility to existing clients (provisioning) and services (consumption) must be preserved.

Figure 3 (right) depicts a plain BPMN flow where the start/end events represent the inbound case, providing the process as a service has a well-defined interface. A new process instance is spawned upon receiving an inbound “request” message on that interface. In turn, the end event terminates the instance and crafts the corresponding outbound “response” message. When compatibly extending that interface to accommodate additional fields, clients (including “parent” processes) may pass on extra data to

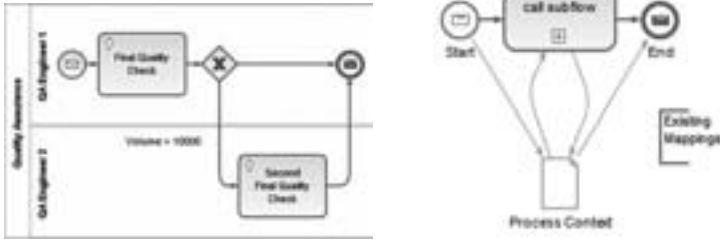


Fig. 3. Usage Extensibility (left) and Field Extensibility (right) Examples

the process. The process may then make use of this data in usage-extended activities. Existing clients remain unaffected, thus, passing (receiving) their inbound (outbound) messages to (from) an extensibility framework which adds (strips off) the extra fields.

Vice versa, the subflow activity constitutes the consumption case where the activity's interface may be field extended in the same manner. Altogether, Field Extensibility is concerned with preserving compatibility despite interface changes. When used in isolation, it does not specify the means to take advantage of additional data fields.

5 Open Research Challenges

In this paper, we introduce the idea of process extensibility but do not yet cover the whole topic exhaustively. In fact, we believe extensibility constitutes a whole new area of BPM research. In this section, we present a research agenda that gives indications for future research on conceptual and technical follow-up topics. Most topics revolve around (1) fully understanding the applicability and limitations of process extensibility and (2) laying its formal and technical foundations:

Extensibility Patterns To set the scene for follow-up research, it is important to gain a comprehensive overview on relevant extensibility use-cases. These use-cases should preferably constitute real-world customization requirements which need to be classified and mapped to extensibility patterns.

Reference Process Conformance Extensions alter the behavior of reference processes which are, in turn, supposed to represent best practices. It is thus necessary to preserve some core characteristics of an extended process. Future work in this area could result in an explicit constraint model for defining extensions for reference processes.

Reference Process Patchability After shipment, a reference process p is solely maintained through patching (cf. Fig. 4, left). The vendor may ship a new version p' that all existing extensions transparently apply. Hence, existing extensions (e_1) implicitly impose compatibility rules which constrain to what extent a patched reference process p' can differ from the predecessor version p . Future research should formulate compatibility rules for reference process patching. That includes providing migration instructions to automatically handle “dangling extensions” that no longer match a patched reference process.

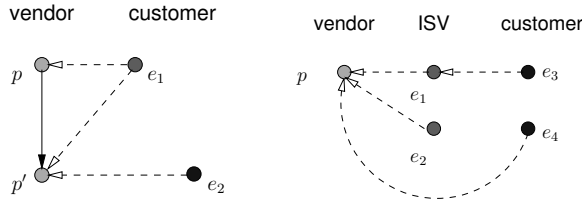


Fig. 4. Reference Process Patchability (left) and Stacked Extensions (right)

Extension Mining Deviations from reference processes may initially not be specified as proper extensions. Instead, end users may also make use of costly ad-hoc changes to gain the required flexibility. To liberate end users from tedious ad-hoc changes, and thus, essentially saving costs, process log mining may be employed to (1) detect “manual” deviations from a reference process original behavior and (2) automatically derive extension definitions.

Extension Point Extension points are part of a reference process and expose its extensible aspects. Future work should develop a concept for specifying extension points, capturing all extension patterns. That may include additional constraints on the extensions that are “plugged in”. Finally, extension points should be self-sufficient such that reference processes could also be shipped as “black box” content, omitting implementation details.

Stacked Extensions In large software rollouts, 3rd party contractors may be involved. For instance, a contractor may be responsible for customizing reference processes through some baseline extensions. The customer itself may further refine these contractor-defined extensions by providing other extensions on top of it. In this way, a transitive extension chain may emerge. Figure 4 (right) depicts a scenario where both a contractor and the customer define extensions atop a reference process p . Customer extensions (e_3 and e_4) can both refer to a contractor extension (e_1) or the reference process directly. Future work needs to devise an extensibility framework architecture that supports these scenarios.

Business Process Outsourcing Both *Software-as-a-Service* and *Cloud Computing* promise significant cost savings through scaling effects. In this regard, *Business Process Outsourcing* has become the corresponding catchphrase for the BPM realm. The idea is to externalize execution of processes to 3rd party hosting providers. In terms of extensibility, one might host the reference process at the vendor side, making invocations to extensions which run on the customer side. Future work should yield an extensibility framework architecture supporting distributed execution environments that tackle challenges like performance, availability, transactionality, failover, authorization, etc.

Authorization Issues Role awareness is a key differentiator of extensibility, as opposed to other process flexibility approaches. Consequently, authorization becomes an issue inasmuch as certain operations (like view, patch, extend, run) may be constrained to certain roles. For instance, the reference process may solely be patched by the vendor, but may be extended on the customer side. More finely grained authorization schemes may be invoked to further constrain the roles that may define

extensions for specific extension points. Altogether, future research should define a comprehensive authorization concept, supporting the fore-mentioned use cases.

Design Time Usability The extensibility approach promises great cost savings over other flexibility approaches. As a prerequisite, BPMS need to include modelling tools to define extensions. These tools need to visualize relevant aspects of the to-be-extended reference process and to define and “wire up” extensions in an easy to comprehend fashion such that the impact of those changes becomes unambiguously evident.

This agenda is by no means complete: our focus is to lay the foundations for practically-oriented extensibility support as part of a BPMS.

6 Related Work

In this paper, *business process extensibility* is positioned as a new area of research in the well-explored field of *process flexibility*. A recent taxonomy in process flexibility [SMR⁺08] identified four approaches to achieving flexibility:

- *flexibility by design* – where a number of alternative pathways are explicitly specified in the process model at design time.
- *flexibility by deviation* – where at run-time an alternative course of action can be taken which differs from the course of action prescribed by the process model.
- *flexibility by underspecification* – where detailed specification of (parts of) the process model is avoided. As mentioned in [SMR⁺08], this category covers both *late modelling* and *late binding*.
- *flexibility by change* – where a process model can be modified after deployment.

BPM systems such as ADEPT1 [RRD03], YAWL [AtH05] (including its Worklet service [AHEA06]), FLOWer [AWG05] and DECLARE [PSSA07] are classified in [SMR⁺08] according to this taxonomy.

Patterns are a useful means to compare the capabilities of different languages/systems and there are two pattern collections in the area of process flexibility that have recently been developed for this purpose. On the one hand, so-called *change patterns* and *change support features* are documented in [WRRM08], while on the other hand the flexibility taxonomy gave rise to a collection of *flexibility patterns* [MAR08]. In [MAR08] it is claimed that the “majority of” the change patterns can be “mapped on” the flexibility patterns. Neither pattern collection addresses the issue of managing the evolution of (reference) process models by vendors and of their counterparts by customers. However, they can be used as a mechanism to operationalize our ideas.

A well-researched problem in the area of dynamic/adaptive workflow is the migration of process instances across different versions of a process model. Consider e.g. early work by Ellis et al. [EKR95] or van der Aalst [Aal01] dealing with changed control-flow dependencies. A comparative overview of correctness criteria used by various approaches is presented in [RRD04]. More recently, Rinderle et al [RMRW08] investigated new, more relaxed, correctness criteria for process migration, taking not only the control flow perspective but also the data perspective into account. Work in

this area could be exploited and extended to deal with (controlled) changes by the vendor, the customer, or both. The last case in particular poses a challenge.

Reference models are models for targeted application domains that incorporate “best practice” [KKR06] methods in these domains. Reference process models serve to capture the procedural aspects of best practices. In the SAP R/3 environment many such models are made available using the Event-driven Process Chain (EPC) notation. As a reference process model may be quite large in order to capture all possible pathways in the various settings in which it may be used, the notion of a *configurable reference process* models was introduced [RA07]. Customizing a configurable reference process model to a particular setting may lead to a model in which many of the pathways were eliminated as they are simply not applicable. Process configuration typically is a one-off activity where there is no provision for further adaptation of the configured model. Additionally, evolution of configurable reference process models has not yet been investigated or even identified as a topic worthy of research.

An approach to tackling challenges dealing with a collection of so-called “process variants” is documented in [HBR08]. It is proposed that for a process variant the change operations that need to be applied to derive it from a base process model are explicitly stored, rather than keeping only the results of these operations. This is an idea that is valuable to the area of business process extensibility as well. The mixture of design-time and run-time considerations as well as the requirement of supporting restricted changes and the propagation of such changes, position the field of business process extensibility uniquely with respect to process flexibility and process configuration.

7 Summary

This paper introduced the notion of process extensibility as a new paradigm for customizing reference process models and managing their evolution over time. The main difference with traditional process flexibility approaches arises from the clear separation of concerns between the reference process owner (vendor) and the owner of extensions thereof (customer). The tension between customer freedom, when it comes to reference model adaptation, and the ability to incorporate with relatively low effort vendor-initiated reference model changes, needs to be carefully balanced. This paper provided an overview of, and motivation for, the notion of business process extensibility, positioned this area with respect to related areas such as process configuration and process flexibility, and identified some of the main unresolved challenges in this area. The limitation of *controlled flexibility* presented in this paper is related to the ability of the vendor to foresee where extensions to a reference model could be needed in future. In fact once extension points are set, they should not be changed to avoid losing synchronization with the customers’ derived models.

References

- [Aal01] W.M.P. van der Aalst. Exterminating the dynamic change bug: A concrete approach to support workflow change. *Inf. Systems Frontiers*, 3(3):297–317, 2001.

- [AHEA06] M. Adams, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Worklets: A service-oriented implementation of dynamic flexibility in workflows. In *Proc. of the 14th Int. Conf. on Cooperative Inf. Systems (CoopIS'06)*, volume 4275 of *LNCS*, pages 291–308. Springer, 2006.
- [Ath05] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.
- [AWG05] W.M.P. van der Aalst, M. Weske, and D. Grünbauer. Case handling: A new paradigm for business process support. *DKE*, 53(2):129–162, 2005.
- [EKR95] C. A. Ellis, K. Keddara, and G. Rozenberg. Dynamic change within workflow systems. In *Proc. of the Conf. on Organizational Computing Systems, COOCS 1995, Milpitas, California, USA, August 13-16, 1995*, pages 10–21. ACM, 1995.
- [FLZ06] P. Fettke, P. Loos, and J. Zwicker. Business process reference models: Survey and classification. In *BPM Workshops*, volume 3812 of *LNCS*. Springer, 2006.
- [GAJVL08] F. Gottschalk, W.M.P. van der Aalst, M.H. Jansen-Vullers, and M. La Rosa. Configurable Workflow Models. *Int. Journal of Coop. Information Systems*, 17(2):177–221, 2008.
- [HBR08] A. Hallerbach, T. Bauer, and M. Reichert. Managing process variants in the process life cycle. In *ICEIS 2008 - Proc. of the Tenth Int. Conf. on Enterprise Information Systems, Volume ISAS-2*, pages 154–161, 2008.
- [KKR06] J. M. Küster, J. Koehler, and K. Ryndina. Improving business process models with reference models in business-driven development. In *BPM 2006 Workshops*, volume 4103 of *LNCS*, pages 35–44. Springer, 2006.
- [MAR08] N. Mulyar, W.M.P. van der Aalst, and N. Russell. Process flexibility patterns. BETA Working Paper Series, WP 251, Eindhoven University of Technology, the Netherlands, 2008. Available at http://fp.tm.tue.nl/beta/publications/working%20papers/Beta_wp251.pdf.
- [PSSA07] M. Pesic, M. H. Schonenberg, N. Sidorova, and W.M.P. van der Aalst. Constraint-based workflow models: Change made easy. In *CoopIS, DOA, ODBASE, GADA, and IS, OTM Confederated Int. Conf. Proc., Part I*, volume 4803 of *LNCS*, pages 77–94. Springer, 2007.
- [RA07] M. Rosemann and W.M.P. van der Aalst. A Configurable Reference Modelling Language. *Information Systems*, 32(1):1–23, 2007.
- [RMRW08] S. Rinderle-Ma, M. Reichert, and B. Weber. Relaxed compliance notions in adaptive process management systems. In *Conceptual Modeling - ER 2008, 27th Int. Conf. on Conceptual Modeling*, volume 5231 of *LNCS*, pages 232–247. Springer, 2008.
- [RRD03] M. Reichert, S. Rinderle, and P. Dadam. Adept workflow management system:. In *BPM 2003*, volume 2678 of *LNCS*, pages 370–379. Springer, 2003.
- [RRD04] S. Rinderle, M. Reichert, and P. Dadam. Correctness criteria for dynamic changes in workflow systems - a survey. *DKE*, 50(1):9–34, 2004.
- [RRKD05] M. Reichert, S. Rinderle, U. Kreher, and P. Dadam. Adaptive process management with ADEPT2. In *ICDE 2005*, volume 3716, pages 1113–1114. IEEE Computer Society, 2005.
- [SMR⁺08] H. Schonenberg, R. Mans, N. Russell, N. Mulyar, and W.M.P. van der Aalst. Towards a taxonomy of process flexibility. In *CAiSE Forum*, pages 81–84, 2008.
- [SN00] A.-W. Scheer and M. Nüttgens. *Business Process Management*, volume 1806 of *LNCS*, chapter ARIS Architecture and Reference Models for BPM. Springer, 2000.
- [WRRM08] B. Weber, M. Reichert, and S. Rinderle-Ma. Change patterns and change support features - Enhancing flexibility in process-aware information systems. *DKE*, 66(3):438–466, 2008.

Process Flexibility: A Design View and Specification Schema

Udo Kannengiesser

NICTA, Locked Bag 9013, Alexandria NSW 1435, Australia, and
School of Computer Science and Engineering, University of New South Wales, Sydney,
Australia
udo.kannengiesser@nicta.com.au

Abstract: This paper proposes a framework of process flexibility based on a view of processes as design objects. It is represented using the function-behaviour-structure (FBS) ontology of designing. The paper shows how the FBS ontology allows extending and generalising recent work on flexibility in engineering design, and how it allows applying this work to processes. The resulting framework provides a comprehensive account of process flexibility that subsumes existing approaches. Finally, the paper presents a specification schema for process flexibility, illustrated using examples of a property valuation process in the Australian lending industry.

1 Introduction

Flexible modelling of processes is a key issue for the effective use of process-aware information systems (PAIS) in dynamic business environments [WSR09]. Factors such as market or strategy changes, technological innovations and new regulations often require modifications of a process. Furthermore, unforeseen events in the immediate environment of the process need to be handled flexibly, such as resource bottlenecks or effects of unexpected human or system errors. PAIS using process models that are too rigidly specified are poorly applicable in real-world contexts and are ultimately rejected by their users.

Research has been concerned with understanding, modelling and implementing the notion of process flexibility. The taxonomies and methods resulting from these efforts address a wide range of aspects of flexibility [PV06, RSS06, DN07, Re07]. However, there is no coherent, comprehensive framework of process flexibility, as most approaches have been developed independently of each other. An attempt to providing such a framework has recently been undertaken by [Sc08], proposing a general taxonomy of process flexibility associated with different realisation approaches.

Flexibility in PAIS is often viewed as a balance between the freedom to change and the need for stability [Re07]. This balance is also an inherent characteristic of designing: Designers aim to change parts of the world through their designs, balancing the use of their individual perception and creativity, and the need to comply with requirements and constraints. A view of process performers as process re-designers has been well described by [Va07]. This paper explores this design view of flexibility, aiming to establish broader, interdisciplinary foundations for understanding and specifying process flexibility. This provides the basis for augmenting rather than replacing existing frameworks of process flexibility.

Section 2 introduces an ontology of designing, the function-behaviour-structure (FBS) ontology, that can be applied to any object of designing, no matter whether this object is a physical product, a software product, or a process. The FBS ontology is then used to extend and generalise recent work on flexibility in engineering design. This provides the basis for a mapping between the design view of flexibility and existing frameworks of process flexibility. Section 3 derives a schema for specifying process flexibility based on the FBS ontology. Examples from the domain of real estate valuation illustrate the use of this schema. Section 4 concludes the paper.

2 A Design View of Process Flexibility

2.1 The Function-Behaviour-Structure View of Designing

Design objects can be modelled using the FBS ontology [GK04, GK07] that has been applied to various instances of design objects, including physical products [GK04], software [Kr05] and processes [GK07].

Structure (S) is defined as a design object's components and their relationships. It can be viewed as the final outcome of a design process. In the domain of physical products, structure comprises the geometry, topology and material of individual components or assemblies. The structure of software consists of abstract constructs such as classes, components and pieces of code. In the domain of processes, structure includes three general classes of components: input, transformation and output [GK07]. The transformation often consists of a set of sub-transformations, some of which can be viewed as "micro-level" mechanisms that represent the "materials" of the transformation. For example, a sequence of activities concerned with logging into an online banking system, and filling out and submitting a funds transfer form can be viewed as a process-centred "material" of a payment transformation [Ka08]. Other "materials" are object-centred; they refer to the agent performing the transformation. Process-centred and object-centred "materials" map onto Dietz' notions of realisation and implementation of a process, respectively [Di06]. Structure encompasses control-flow, data, resource, and task views of a process [Ka08].

Behaviour (B) is defined as the attributes that can be derived from a design object's structure. They provide criteria for comparing and evaluating different design objects. An example of a physical product's behaviour is "weight", which can be derived (or measured) from the product's structure properties of material and spatial dimensions. Behaviour of software (e.g., a text editor) includes its response time for visualising user input. It can be derived from software structure and its interaction with the operating environment. Typical behaviours of processes include speed, cost, precision and accuracy. They can be derived from process structure; for example, speed can be derived from (time-stamped) input and output.

Function (F) is defined as a design object's teleology ("what it is for"). This notion is independent of the common distinction between "functional" and "non-functional" properties; it comprises both as they describe the design object's usefulness for a stakeholder (or "using system" [Di06]). It should not be confused with the concept of "transfer function". Function is ascribed to behaviour by establishing a teleological connection between a human's goals and measurable effects of the design object. There is no direct connection between function and structure. The particular functions of a design object are ontologically independent of whether the design object's structure is conceptualised as a physical product, a software product or a process. For example, the functions "wake people up", "be reliable" and "be punctual" may be ascribed to relevant behaviours of a mechanical alarm clock (i.e., a physical product), a virtual alarm clock (i.e., software), or a sequence of activities (i.e., a process).

From a high-level perspective, designing can be viewed as decision making. This view implies the existence of choices [Ge94] that can be represented as alternative values for the variables of the design. The set of all design variables and their ranges of values form what is called the design state space, i.e. the space of all possible designs. The design state space is partitioned into three subspaces: function state space, behaviour state space, and structure state space, as shown in Figure 1. The three subspaces are interconnected through the designer's compiled knowledge of qualitative and quantitative relationships between function, behaviour and structure. These relationships are the basis for modelling designing as an activity that aims to produce structure that exhibits suitable behaviour to which desired function can be ascribed.

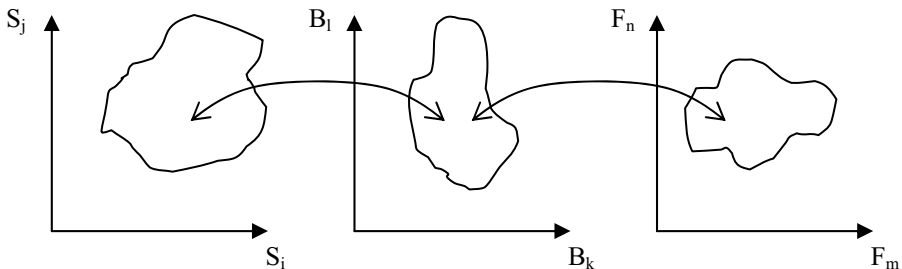


Figure 1: Function, behaviour and structure state spaces, and their interconnections

The notion of a design state space allows understanding designing through the use of spatial metaphors. Selecting values for a set of design variables can be described as a process of moving through the design state space. This model of designing is often referred to as *search* [Ge94]. However, most designing involves more than moving through a well-defined space of known design alternatives. It also involves generating the space in terms of design variables and their ranges of values. This can involve discarding some previously expected variables or ranges of values, leading to a shift of the design state space. The notion that addresses these changes is called *exploration* [Ge94]. Changes may affect all three subspaces, and changes of one subspace may lead to changes of a subspace connected to it. For example, a change of the structure state space may lead to changes of the behaviour state space. This, in turn, may lead to subsequent changes of the function state space and/or the structure state space.

Expectations about the design problem are fundamental in distinguishing exploration from search. Exploration reflects changed expectations as the designer learns more about the design problem by interacting with it [Sc83]. In contrast, the notion of search reflects unchanged expectations of the design (state space). Some of the initial design expectations are formulated through explicitly stated requirements. Others arise from the designer's understanding of the design object's socio-technical environment across the life cycle. The possible change of a design state space from its inception to a later point in time is presented conceptually in Figure 2. The increasing size of the design state space is to indicate that a great deal of the knowledge required to produce a design is constructed during designing [LS93].

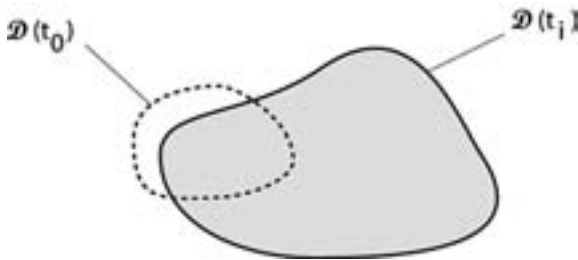


Figure 2: A design state space (\mathcal{D}) changes between the initial time t_0 and a later time t_i

The application of the state space concept in PAIS has commonly had a more narrow focus. It is usually understood only as a representation of the set of possible changes in the world that may occur during the execution of a process instance, not as a representation of the set of all possible process designs. Recent work on business rules in process models [Di08] can be viewed as expanding the scope of state space models of processes. However, these approaches are limited to the notion of a structure state space, and do not cover behaviour and function state spaces.

Design state spaces can be represented in many ways. The most common representations include enumerations of states (e.g., a set of alternative product modules or process fragments), generative representations (e.g., grammars), constraints, and abstraction (e.g., using types defined in domain ontologies).

2.2 Generalising an Engineering Design Approach to Flexibility

Flexibility has been a popular concept in many areas within engineering design. Similar to process flexibility, it is understood here as the ability of a product or system to handle change. However, more precise definitions of this notion are often missing [SHN03]. One of the most comprehensive approaches to defining and characterising flexibility in engineering design has recently been proposed by researchers from the MIT Engineering Systems Division [RRH08]. This Section provides an overview of relevant concepts of this work, and expands and generalises them using the FBS ontology. [RRH08] propose two aspects of flexibility of a design object: change effects, and change mechanisms.

Change effects characterise the difference between the states of a design object before and after its change. States are described in terms of variables and values that may refer to any aspect of the design object, including function, behaviour and structure. There are three categories of change effects: robustness, scalability, and modifiability.

Robustness is the ability to maintain the design object's required functions without changing its structure, despite the presence of changes affecting the object's internal or external environment [RRH08]. For example, a car may achieve its function of transportation without changing its design, despite internal changes such as tire abrasion or external changes such as altered road conditions. Robustness handles change by being insensitive to it. In fact, it has been understood as a concept that is related but quite distinct from flexibility [SHN03].

Scalability is the ability to vary the design object's state in terms of the values of its variables [RRH08]. For example, varying the length, width and height of a mobile phone is a scalable change of structure. Varying the speed of a central processing unit is a scalable change of behaviour. And varying the reliability of an alarm clock is a scalable change of function. Scalability is captured in the state space representation of designing as either search (if the change remains within state space boundaries) or exploration (if the change involves crossing a state space boundary in terms of ranges of values).

Modifiability is the ability to vary the design object's state in terms of its variables [RRH08]. For example, the addition of a DVD burning module to a computer is a modifiable change of structure. Changing a car's petrol consumption rate to rapeseed oil consumption rate is a modifiable change of behaviour. And augmenting a mobile phone with the ability to play MP3 files is a modifiable change of function. Modifiability is captured in the state space representation of designing as exploration via changing the set of variables.

Variations of function, behaviour and structure rarely occur in isolation of each other. As outlined in Section 2.1, a change of one subspace often leads to changes of other subspaces. The specific ways in which a change propagates across the subspaces depends not only on given requirements and constraints but also on the individual expertise and interpretations of the designer. We can view this "design freedom" as an additional dimension of flexibility, embedded within the notion of change effects.

Change mechanisms represent different ways of achieving the desired change effects. [RRH08] propose the number of possible change mechanisms, filtered by a subjective acceptability threshold for their “cost”, as a basis for quantifying flexibility. Here, “cost” is an aggregated measure for the consumption of various resources including time and money.

We can expand the notion of change mechanisms by defining three categories, Figure 3: design goal achievement, design realisation, and design assessment.

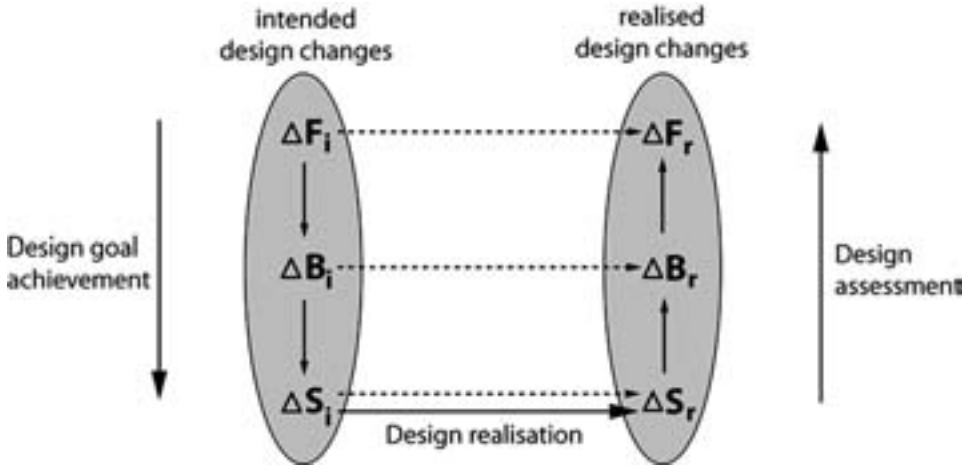


Figure 3: Three categories of change mechanisms: design goal achievement, design realisation, and design assessment.

Design goal achievement is the ability to vary the activities and resources required for transforming intended changes of function (ΔF_i) into intended changes of structure (ΔS_i) via intended changes of behaviour (ΔB_i). For example, besides generating design changes from scratch, one may have the option of reusing previous design knowledge captured in patterns, best practices, rationale, case bases, prototypes or other forms of representation. Each of these options requires different technologies and user skills.

Design realisation is the ability to vary the activities and resources required for transforming an intended change of structure (ΔS_i) into a realised change of structure (ΔS_r). This includes the allocation of agents (machines or people), their coordination and any setup tasks required (for example, programming, instructing and training). It also includes strategies for preventing or mitigating issues, such as downtime and obsolete work items arising from changes in the realisation.

Design assessment is the ability to vary the activities and resources required for monitoring, analysing and validating the success (for example, the consistency, correctness and efficiency) of the change. This includes the methods and tools available for deriving changes of behaviour (ΔB_r) from a realised change of structure (ΔS_r), and ascribing changes of function (ΔF_r) to these changes of behaviour.

2.3 Applying the Design View to Process Flexibility

The three categories of change effects (robustness, scalability, and modifiability) and the three categories of change mechanisms (design goal achievement, design realisation, and design assessment) can be mapped onto existing research in flexible PAIS.

Robustness maps onto “flexibility by design” [Sc08] or “flexibility by definition” [SSO01] that is the ability to include multiple execution paths in the process model at design time. They represent different ways of dealing with anticipated variations and exceptions occurring in the execution environment. The different paths are selected at runtime for individual process instances.

Scalability maps onto two categories of process flexibility proposed by [Sc08] that imply the existence of expected choices. One is “flexibility by deviation” [Sc08] that is the ability of a process instance to deviate from the original process model (type) without altering it. It encompasses only changes in the execution sequence of tasks, not the tasks themselves. We conceptualise the ordering relationships that determine the execution sequence as a set of interrelated “ports” of the tasks [Go08]. Specifically, every task has variables for their “inflow ports” and “outflow ports”, and the values of these variables are pointers to other tasks. Changing the relationships can then be viewed as varying the values of structure variables. The other category of process flexibility corresponding to scalability is “flexibility by underspecification” [Sc08] or “flexibility by templates” [SSO01] via late binding of process fragments to a placeholder. This category of process flexibility is the ability to execute an incomplete process model by completing it at runtime, via selection from a pre-defined set of process fragments. The fragments can be represented as structure variables with Boolean values. A process fragment with the value “false” means that this fragment is currently not selected. Changing the value to “true” corresponds to selecting it to instantiate the placeholder. Potential subjects of scalable change include not only control flow but also other aspects of process structure [RSS06].

Scalable changes of process function and process behaviour are not included in existing work on process flexibility. Examples include improving maintainability of a process (a scalable change of function), and reducing the cost of a process (a scalable change of behaviour).

Modifiability maps onto two categories of process flexibility proposed by [Sc08] that imply a shift of expectations. One is “flexibility by underspecification” [Sc08] via late modelling, which is the ability to construct a new process fragment for a placeholder. It is best thought of as the generation of a new variable to be introduced in the structure state space of the process. The other category mapping onto modifiability is “flexibility by change” [Sc08], which is the ability to modify a process at runtime, in response to unforeseen circumstances in the execution environment. Here, changes represent new tasks being introduced and/or removed, affecting process instances and/or process types. By representing every task as a structure variable, we can model these changes as modifications of the structure state space. Potential subjects of modifiable change include not only control flow but also other aspects of process structure [RSS06].

Modifiable changes of process function and process behaviour are not included in most existing work on process flexibility. Examples include considering the waste production of a manufacturing process in addition to other process attributes (a modifiable change of behaviour), and changing the goal of a transportation process from “people transportation” to “cargo transportation” (a modifiable change of function). There is work on using variations of quality goals to generate different configurations of process structure [e.g., LYM07].

Design goal achievement is addressed by the range of methodologies for process design. Methodologies differ in their notations, their coverage of different process views, their technological support, and their ease of use. A number of existing technologies, including ADEPT1, YAWL, FLOWer and Declare, have been evaluated by [Sc08] regarding their support for the different change effects.

Design realisation subsumes migration strategies for running process instances, and mechanisms for version, access and concurrency control, which are described in [WRR08]. Design realisation also includes methods and technologies for communication and “setup” (instructing, training, etc.).

Design assessment includes methods and technologies for analysing correctness, consistency, efficiency, traceability, usability and other process quality attributes, which are described in [WRR08].

3 Specifying Process Flexibility

3.1 A Schema for Process Flexibility

The framework presented in Section 2 can be used as the basis for a schema for flexible process specification, whose central notion is the design state space with its three subspaces for function, behaviour and structure. We have presented this notion as the set of all possible designs based on the expectations and experience of the individual designer. When we adopt a prescriptive stance, a design state space becomes the set of all “permitted” designs according to specifications given to the process designer. Here, the boundaries of the design state space, both in terms of the specified set of variables and their ranges of values, represent requirements that may be socially enforceable. In fact, the specified design state space represents a “normative restriction of design freedom” [Di06].

On the other hand, as shown in Figure 2, parts of the initial specification of a design state space may be relaxed over the course of designing, resulting in a new design state space with modified boundaries. Therefore, different degrees of “normative strength” of the state space boundaries should be made explicit to specify what parts of the space may be changed and what parts must not. This can be realised by associating individual design variables and their ranges of values with modality attributes such as “mandatory” and “optional” (or a finer-grained set of attributes such as proposed by [BS06]).

Table 1 shows the resulting specification schema. The columns represent FBS level, state space, and modality. The contents of the white boxes in this Table are suggested approaches to representing these notions in a way that is easy to comprehend for readers of this paper. This is the reason why we use abstraction for representing the structure state space, despite the lack of well-defined, formal domain ontologies for most PAIS. Abstraction is also likely to be useful in phases of requirements elicitation and process definition, where domain experts negotiate a common, high-level view of the process. More formal representations, such as declarative (constraint-based) notations of process structure, are certainly needed in the later phases of process execution and process analysis.

Table 1: A specification schema for process flexibility

| FBS level | State space | Modality |
|-----------|--------------------------|-----------------------|
| F | Enumeration of functions | [mandatory, optional] |
| B | Constraints | [mandatory, optional] |
| S | Abstraction | [mandatory, optional] |

The schema can be used for specifying not only change effects but also change mechanisms. This is because change mechanisms can themselves be viewed as design objects that can be described using the FBS ontology. Change effects and change mechanisms are then represented in two separate sets of specifications but using the same schema. However, in practice only few aspects of change mechanisms are specified explicitly. These are typically aspects related to design realisation, rather than design goal achievement and design assessment. As outlined in Section 2.1, these aspects can be captured as “materials” of process structure.

3.2 Examples

This Section demonstrates the use of the specification schema for a number of (sub-) processes in the context of property valuation (short: valuation) in the Australian lending industry. Figure 4 shows a simplified model of the valuation process in BPMN (Business Process Modeling Notation; see www.bpmn.org). The process starts when the valuation company receives a request from a lender (e.g., a bank) to assess the market value of a specific property. An employee (called the “valuer”) is then assigned to perform the valuation by inspecting the property and preparing a valuation report that contains the estimated market value of the property. After that, the valuation report is sent to the lender, and, concurrently, an invoice is sent. Upon receipt of payment, the valuation process terminates. This process model is assumed to be fixed; we will specify flexibility only for some of the sub-processes in this model.

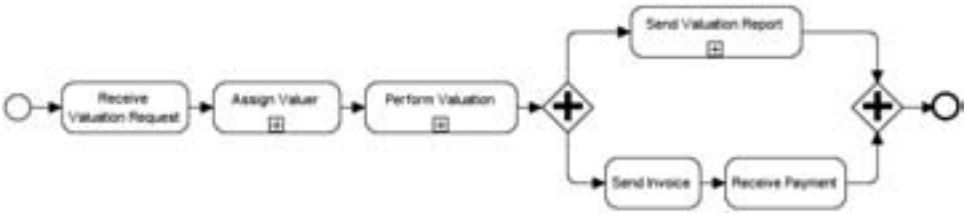


Figure 4: Top-level model of a property valuation process

Figure 5 shows the details of the sub-process “Perform Valuation”. It includes multiple paths that handle cases in which valuation fees need to be renegotiated due to complicated site conditions, such as irregular building shapes or slopes. This is an example of robustness, as the designed process structure is insensitive to changes in (external) site conditions.



Figure 5: Sub-process “Perform Valuation”

Flexibility within the valuation process can be specified using annotations that are structured according to the schema in Table 1. Figure 6 shows three examples.

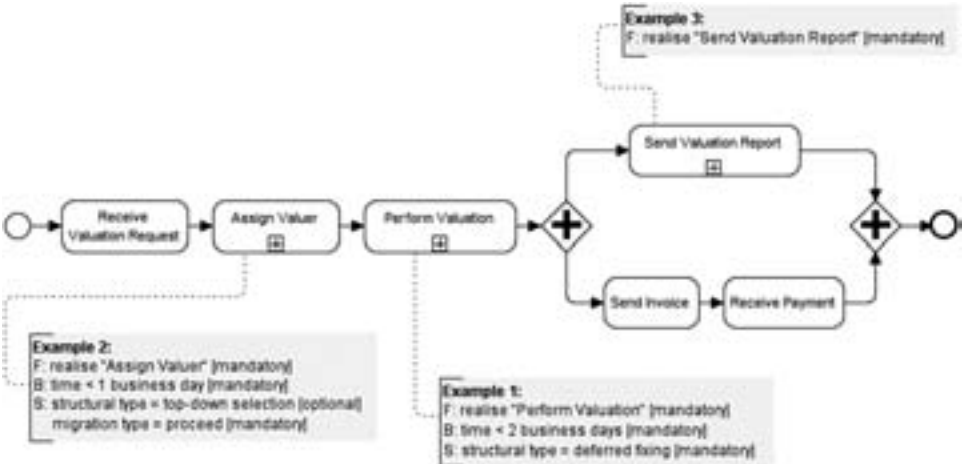


Figure 6: Examples of specifications of sub-process flexibility within the valuation process

Example 1 in Figure 6 is a flexible specification of the sub-process “Perform Valuation”. Here, scalable changes of process behaviour are allowed within the mandatory range of values specified for “time”. The set of process structures that can produce these variations are specified as a mandatory “structural type” referencing “deferred fixing”, which is an exception-handling pattern proposed by [Le08]. The basic idea behind this pattern is that an exceptional situation (here, the complicated site conditions) is identified and recorded, but dealt with (here, by renegotiating fees) later in the process. This abstract description leaves room for various changes in the execution sequence, all of which are scalable changes of process structure. One instance of process structure consistent with this specification was shown in Figure 5. Another instance is shown in Figure 7; here, the position of the fee renegotiation activity within the sub-process is altered. The specification in this example does not constrain the change mechanisms that can be chosen to realise the change effect.

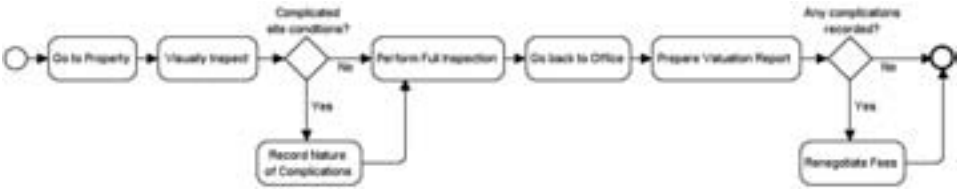


Figure 7: Sub-process “Perform Valuation”, alternative process structure

Example 2 in Figure 6 is a flexible specification of the sub-process “Assign Valuer”. Behaviour is again specified as a mandatory range of values for “time”. Structure includes an optional “structural type” referencing a “top-down selection” procedure such as shown in Figure 8. Suppose we want to drastically reduce the time behaviour to, say, 1 hour instead of 1 business day. This scalable change of behaviour requires a change of structure that is likely to exceed the limits defined in the specification of structure. Since the modality attribute of the structure is “optional” rather than “mandatory”, modifiable changes can be made resulting in a process structure such as shown in Figure 9. Here, the “top-down selection”-type is substituted with a “bidding”-type process structure that can more quickly identify those potential valuers that are currently located near the property to be valued. This process is inspired by the way taxi companies dynamically assign incoming customer requests to specific drivers. This change of process structure requires further changes such as the development and implementation of appropriate information and communication technologies (not shown in the BPMN model). The mechanism for design realisation is specified as a “migration type” that refers to a mandatory “proceed” strategy [Sc08] for migrating existing process instances.



Figure 8: Sub-process “Assign Valuer”, using a “top-down selection”-type process structure



Figure 9: Sub-process “Assign Valuer”, using a “bidding”-type process structure

Example 3 in Figure 6 is a flexible specification of the sub-process “Send Valuation Report”. Here, only the principal function is specified, leaving a great deal of freedom for designing the sub-process. Suppose the designer wants to make a modifiable change of function by including an additional function “provide interoperable data”. This is a very realistic scenario as the Australian lending industry is moving towards interoperable, straight-through processing based on the standard Credit Application Language (CAL) currently being defined by the LIXI consortium (Lending Industry XML Initiative; see www.lixi.org.au). The additional function may lead to a new behaviour variable termed “LIXI compliance”, and new structure variables including “output file type = XML” and “output vocabulary type = CAL”. These are modifiable changes of behaviour and structure. The example does not specify any constraints on how these changes are to be realised in terms of change mechanisms.

4 Conclusion

A design view of process flexibility leverages a characteristic that is inherent in the nature of designing: the capacity to operate within a space of alternatives that is generated based on not only a set of requirements but also the designer’s individual understanding of the problem. The framework presented in this paper expands and generalises a recent approach from engineering design, using a domain-independent ontology of designing. Current approaches to modelling process flexibility fit in this framework, but fall short of covering essential aspects including function and behaviour. These aspects capture “what should be done without specifying how it should be done” [PV06] in a more comprehensive way than existing declarative approaches that are limited to process structure. The design view provides a unifying framework that brings together different research streams in flexible PAIS, most of which can be categorised as focusing on either change effects or change mechanisms.

The proposed specification schema can be used to define the flexibility of a process at different design-ontological levels and with different degrees of “normative strength”. It supports the view of stakeholders as designers or re-designers of the process, while constraining their design activities using a necessary and sufficient set of specifications. The examples in this paper demonstrate a range of process designs that can be generated based on different process specifications. One issue that is not addressed here is the need to move from one state space representation to another. The light-weight approach we utilised for representing structure state spaces is useful for high-level communication among domain experts; however, for process implementation a more formal approach is needed based on well-defined domain ontologies and executable process notations. Progress in this area is likely to draw on research in process patterns [Va03], configurable process models [Go08] and semantic business process management [We07].

There are opportunities to expand the design view of process flexibility, using further analogies from engineering design. One research direction may focus on the qualitative relationships between function, behaviour and structure state spaces. Capturing them can guide process designers realising desired changes of function through appropriate changes of behaviour and structure. For example, research in product family design maps different types of product families (that can be modelled as sets of scalable or modifiable behaviours and structures) onto strategies for targeting different market segments (that can be modelled as sets of scalable or modifiable functions) and onto different manufacturing paradigms (that can be modelled as design realisation options) [MF07]. Research in PAIS is likely to benefit from further investigation of these analogies, as they allow tapping into well established methodologies of flexibility from various domains.

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

Bibliography

- [BS06] Borch, S.E.; Stefansen, C.: On Controlled Flexibility. In (Latour, T.; Petit, M., Eds.): Proceedings of Workshops and Doctoral Consortium, The 18th International Conference on Advanced Information Systems Engineering – Trusted Information Systems, Namur University Press, Namur, 2006, pp. 121-126.
- [DN07] Daoudi, F.; Nurcan, S.: A Benchmarking Framework for Methods to Design Flexible Business Processes. *Software Process: Improvement and Practice*, 12(1), 2007, pp. 51-63.
- [Di06] Dietz, J.L.G.: *Enterprise Ontology: Theory and Methodology*, Springer-Verlag, Berlin, 2006.
- [Di08] Dietz, J.L.G.: On the Nature of Business Rules. In (Dietz, J.L.G.; Albani, A.; Barjis, J., Eds.): *Advances in Enterprise Engineering I*, LNBIP 10, Springer-Verlag, Berlin, 2008, pp. 1-15.
- [Ge94] Gero, J.S.: Towards a Model of Exploration in Computer-Aided Design. In (Gero, J.S.; Tyugu, E., Eds.): *Formal Design Methods for CAD*, North-Holland, Amsterdam, 1994, pp. 315-336.
- [GK04] Gero, J.S.; Kannengiesser, U.: The Situated Function-Behaviour-Structure Framework. *Design Studies*, 25(4), 2004, pp. 373-391.
- [GK07] Gero, J.S.; Kannengiesser, U.: A Function-Behavior-Structure Ontology of Processes. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 21(4), 2007, pp. 379-391.
- [Go08] Gottschalk, F. et al.: Configurable Workflow Models. *International Journal of Cooperative Information Systems*, 17(2), 2008, pp. 177-221.
- [Ka08] Kannengiesser, U.: Subsuming the BPM Life Cycle in an Ontological Framework of Designing. In (Dietz, J.L.G.; Albani, A.; Barjis, J., Eds.): *Advances in Enterprise Engineering I*, LNBIP 10, Springer-Verlag, Berlin, 2008, pp. 31-45.

- [Kr05] Kruchten, P.: Casting Software Design in the Function-Behavior-Structure Framework. *IEEE Software*, 22(2), 2005, pp. 52-58.
- [Le08] Lerner, B.S. et al.: Exception Handling Patterns for Processes. In (Garcia, A. et al., Eds.): *Proceedings of the 4th International Workshop on Exception Handling*, Atlanta, GA, 2008, pp. 55-61.
- [LS93] Logan, B.; Smithers, T.: Creativity and Design as Exploration. In (Gero, J.S.; Maher, M.L., Eds.): *Modeling Creativity and Knowledge-Based Creative Design*, Lawrence Erlbaum, Hillsdale, 1993, pp. 139-175.
- [LYM07] Lapouchnian, A.; Yu, Y.; Mylopoulos, J.: Requirements-Driven Design and Configuration Management of Business Processes. In (Alonso, G.; Dadam, P.; Rosemann, M., Eds.): *Business Process Management*, LNCS 4714, Springer-Verlag, Berlin, 2007, pp. 246-261.
- [MF07] Maier, J.R.A.; Fadel, G.M.: A Taxonomy and Decision Support for the Design and Manufacture of Types of Product Families. *Journal of Intelligent Manufacturing*, 18(1), 2007, pp. 31-45.
- [PV06] Pesic, M.; Van der Aalst, W.M.P.: A Declarative Approach for Flexible Business Processes Management. In (Eder, J. et al., Eds.): *BPM 2006 Workshops*, LNCS 4103, Springer-Verlag, Berlin, 2006, pp. 169-180.
- [Re07] Regev, G.; Bider, I.; Wegmann, A.: Defining Business Process Flexibility with the Help of Invariants. *Software Process: Improvement and Practice*, 12(1), 2007, pp. 65-79.
- [RRH08] Ross, A.M.; Rhodes, D.H.; Hastings, D.E.: Defining Changeability: Reconciling Flexibility, Adaptability, Scalability, Modifiability, and Robustness for Maintaining System Lifecycle Value. *Systems Engineering*, 11(3), 2008, pp. 246-262.
- [RSS06] Regev, G.; Soffer, P.; Schmidt, R.: Taxonomy of Flexibility in Business Processes. In (Latour, T.; Petit, M., Eds.): *Proceedings of Workshops and Doctoral Consortium, The 18th International Conference on Advanced Information Systems Engineering – Trusted Information Systems*, Namur University Press, Namur, 2006, pp. 90-93.
- [Sc83] Schön, D.A.: *The Reflective Practitioner: How Professionals Think in Action*, Harper Collins, New York, 1983.
- [Sc08] Schonenberg, H. et al.: Process Flexibility: A Survey of Contemporary Approaches. In (Dietz, J.L.G.; Albani, A.; Barjis, J., Eds.): *Advances in Enterprise Engineering I*, LNBIP 10, Springer-Verlag, Berlin, 2008, pp. 16-30.
- [SHN03] Saleh, J.H.; Hastings, D.E.; Newman, D.J.: Flexibility in System Design and Implications for Aerospace Systems. *Acta Astronautica*, 53(12), 2003, pp. 927-944.
- [SSO01] Sadiq, S.; Sadiq, W.; Orłowska, M.: Pockets of Flexibility in Workflow Specification. In (Kunii, H.S.; Jajodia, S.; Solvberg, A., Eds.): *Conceptual Modeling – ER 2001*, LNCS 2224, Springer-Verlag, Berlin, 2001, pp. 513-526.
- [Va03] Van der Aalst, W.M.P. et al.: Workflow Patterns. *Distributed and Parallel Databases*, 14(3), 2003, pp. 5-51.
- [Va07] Van Aken, J.E.: Design Science and Organization Development Interventions: Aligning Business and Humanistic Values. *Journal of Applied Behavioral Science*, 43(1), 2007, pp. 67-88.
- [We07] Wetzstein, B. et al.: Semantic Business Process Management: A Lifecycle Based Requirements Analysis. In (Hepp, M. et al., Eds.): *Semantic Business Process and Product Lifecycle Management. Proceedings of the Workshop SBPM 2007*, Innsbruck, Austria, 2007, pp. 1-10.
- [WRR08] Weber, B.; Reichert, M.; Rinderle-Ma, S.: Change Patterns and Change Support Features – Enhancing Flexibility in Process-Aware Information Systems. *Data & Knowledge Engineering*, 66(3), 2008, pp. 438-466.
- [WSR09] Weber, B.; Shazia, S.; Reichert, M.: Beyond Rigidity – Dynamic Process Lifecycle Support. *Computer Science – Research and Development*, 23(2), 2009, pp. 47-65.

Access Control for Monitoring System-Spanning Business Processes in Proviado

Sarita Bassil¹, Manfred Reichert², Ralph Bobrik³, Thomas Bauer⁴

¹Computer Science Department, Marshall University, USA

²Institute of Databases and Information Systems, Ulm University, Germany

³Detecon AG, Switzerland

⁴Group Research and Advanced Engineering, Daimler AG, Germany

Abstract: Integrated process support is highly desirable in environments where data related to a particular (business) process are scattered over distributed, heterogeneous information systems (IS). A process monitoring component is a much-needed module in order to provide an integrated view on all these process data. Regarding process data integration, access control (AC) issues are very important but also quite complex to be addressed. A major problem arises from the fact that the involved IS are usually based on heterogeneous AC components. For several reasons, the only feasible way to tackle the problem of AC at the process monitoring level is to define access rights for the process monitoring component, hence getting rid of the burden to map access rights from the IS level. This paper discusses requirements for AC in process monitoring, which we derived from our case studies in the automotive domain. It then presents alternative approaches for AC: the view-based and the object-based approach. The latter is retained, and a core AC model is proposed for the definition of access rights that meet the derived requirements. AC mechanisms provided within the core model are key ingredients for the definition of model extensions.

1 Introduction

In order to streamline their way of doing business, today's companies are dealing with a number of processes involving different domains, organizations, and groups. As discussed in [BRB05], an integrated process support is highly desirable in such an environment where data (e.g., audit trails and reports) related to a particular process (instance), and with different degrees of sensitivity, are often scattered over heterogeneous information systems (IS) (cf. Fig. 1). A process monitoring component is a much-needed module in order to provide an integrated view on all these data [JHBK04, Mue01]. Despite its importance, many existing process-aware IS [Wes07] do not offer such a component. Specifically, a process monitoring component is responsible for displaying the status of process instances, for dispatching specific activities to corresponding actors, and so on.

Different user groups or roles (e.g., technicians, managers) usually have different perspectives over processes and related data. Therefore, adequate views need to be provided [BRB07]. This is of particular importance when dealing with complex, long-running busi-

ness processes with dozens up to thousands activities. In the context of process data integration and process monitoring [JHBK04], in addition, access control (AC) issues are very important to be addressed. However, a major problem is that involved IS are usually based on different AC components implying facts such as 1) heterogeneity regarding the meta-models based on which organizational models and related access rights are defined (e.g., users/groups and actors/roles), 2) different notions for the same entity/entity type (e.g., user and actor), and 3) non-registration of particular user(s) in all of the involved IS.

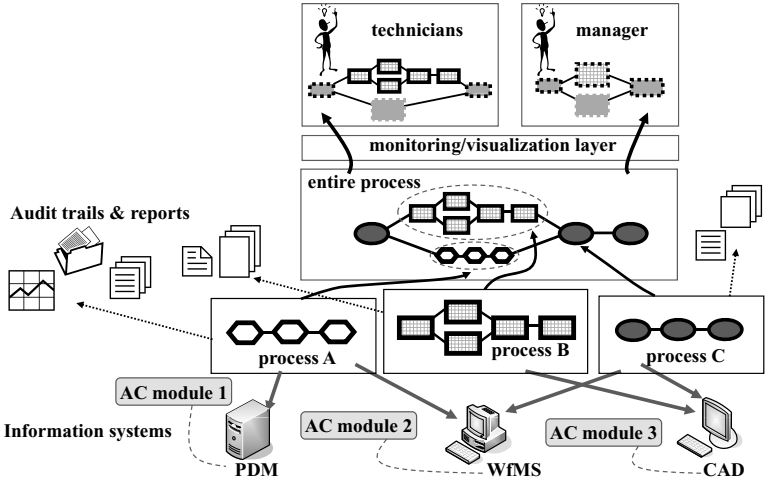


Figure 1: Process Data Integration with Multiple Perspectives

To preserve integrity of AC information, AC constraints applied at the process monitoring level should be consistent with the constraints set out by the different IS. However, it has turned out that the integration of heterogeneous AC components is difficult to achieve for several reasons: 1) Access rights are not always explicitly described, but might be “hard-coded”, and hence difficult to retrieve; 2) AC modules do not always provide interfaces (i.e., APIs) in order to facilitate the access to information about AC rules (“black-box” AC modules); and 3) Rights at the IS level mainly deal with process definition and execution, and have been not designed for the monitoring of process data by different users. Process definition and execution require administration rights, permissions to create new instances, delegation rights, rights to work on specific activities, and rights to change processes [WRWR05]. By contrast, monitoring requires rights to visualize specific process activities, to display specific activity attributes, or to show different abstractions on a process (cf. Fig. 2a+b). Taking this into account, the only feasible way to tackle the problem of AC at the process monitoring level is to (re-)define AC rights for the process monitoring component, hence getting rid of the burden to inherit AC rights from the IS level. Of course, if possible, existing AC rights at the IS level should be automatically mapped to the ones at the process monitoring level, but we cannot assume this in general. Explicitly, specifying AC rights at the monitoring level also makes it possible to define them at a finer-grained level when compared with what is already defined at the IS level.

This paper discusses requirements relevant for the definition of such AC rights. These requirements have resulted from case studies we conducted in the automotive domain.¹ We propose approaches for AC, mainly a *view-based* and an *object-based* one. The retained solution (i.e., the object-based approach) is used as backbone in order to provide a comprehensive core AC model. This model allows for the (compact) definition of AC rights at a fine-grained level. Moreover, AC rights are meant to meet the spectrum of confidentiality possibly defined on process data. Proposed AC mechanisms will be key ingredients in future definitions of extended AC models for process monitoring.

Section 2 discusses basic notions by distinguishing between model and instance level. Section 3 exposes the major requirements identified. Two alternative approaches for AC are studied and compared in Section 4. In Section 5, we introduce our logical AC model. Section 6 discusses related work and Section 7 concludes with a summary and an outlook.

2 Basic Considerations

Generally, we distinguish between model and instance level (cf. Fig. 2). The former gathers different kinds of enterprise models such as organizational models, functional models, data models, IT-system models, and process models. Each of the first four models gives input to the process model defined as a set of one or more linked activities, which collectively realize a business objective. Specifically, these activities are carried out, in a coordinated way, by different processing entities (incl. humans and software systems) to reach a goal, such as changing the design of a car, delivering merchandise, or operating a patient. User- and pre-defined attributes may be associated with process models or activities (e.g., costs, needed resources). Examples of frameworks supporting the integrated modeling of the different enterprise aspects include ARIS and Adonis.

In Proviado [BRB05, BBR06, BRB07], at the *model level*, we focus on the secure visualization of data related to a particular process model. Other kinds of models have not been considered for visualization yet, but will be added later on. Different types of data may be involved in a process model such as process relevant data and application data [Wes07]. We are particularly interested in providing a secure way to visualize application data. These data are in general strictly managed by the application(s) supporting the process model. At the *instance level*, we focus on the secure monitoring of running process instances. A process instance is defined as the representation of a single enactment of a process model (i.e., a concrete business case) [Wes07]. Concepts such as user worklists (i.e., lists of work items derived from process instance activities), activity execution state (e.g., *Running*), and activity execution cost are associated with the instance level.

At model and instance levels, different kinds of rights need to be defined; e.g., administration rights, data access rights, permission to create instances from a given process model, rights to execute a particular work item, or delegation rights. At the model (instance) level, the visualization (monitoring) of user-adapted views derived from specific process models

¹ In the Proviado project [BRB05, BBR06, BRB07], we are aiming to propose a solution for visualizing in a secure way data related to a particular process or to a collection of processes.

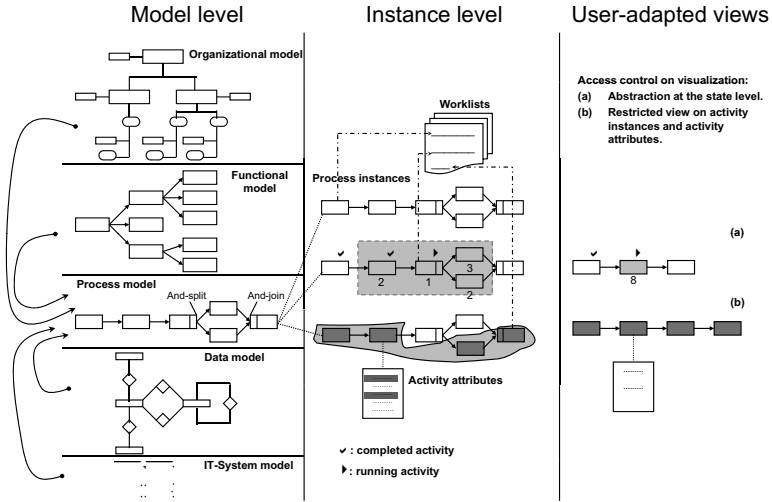


Figure 2: Basic Considerations

(instances) is required. These views must take into account access rights of the involved user. Access rights may be defined on different aspects related to the model and instance levels; e.g., process model, activity, process instance, activity instance, data elements, pre- and user-defined attributes, attribute current value, and attribute history.

3 Access Control Major Requirements

We conducted case studies in the automotive domain in which we studied processes like *car engineering*, *change management* (cf. Fig. 3a) and *release management*. As fruit of these case studies, we derived major requirements for AC in process monitoring.

Requirement 1 (Definition of AC rights at a fine-grained level). AC rights for process monitoring should meet the spectrum of confidentiality defined on data related to a particular process. Moreover, they should be definable on different aspects/objects of the model and instance levels (e.g., the process itself and its activities, attributes, and data elements).

- *Requirement 1.1 (Meeting a spectrum of confidentiality).* A distinction should be made between at least three levels of confidentiality: a first level in which all available information can be accessed, a second one where only high-level information can be accessed, and a third one where no information is available at all. Considering the process of managing change requests (cf. Fig. 3a), for example, we may think about a (pre-defined) attribute e.g., *activity cost* associated with a specific activity (e.g., *generate expertise*). Such an activity may require a “two days by person” cost to be accomplished. One may have the right to access this information (i.e., the exact value of the attribute), to access abstracted information such as “less than one week (i.e., less than five days by person)”, or to access nothing.

The spectrum of confidentiality may also be restricted to only two levels: “give” or “don’t give information”. In change management for example, an external partner may design part of the car; internally, a verification of this component may be done before it is integrated with the overall design of the car. The external partner might or might not have the right to know about the *existence* of the verification activities.

- *Requirement 1.2 (AC rights definable on different objects of the model / instance levels).* We define “object” as entity of a process model / instance; e.g., an *expertise document* produced as output of a generate expertise activity is considered as data object. The generate expertise activity itself as well as the change request (CR) process model are considered as two different objects. Moreover, a group of objects is also an object; e.g., AC rights may be defined 1) on all running CR process instances, or 2) on specific ones.

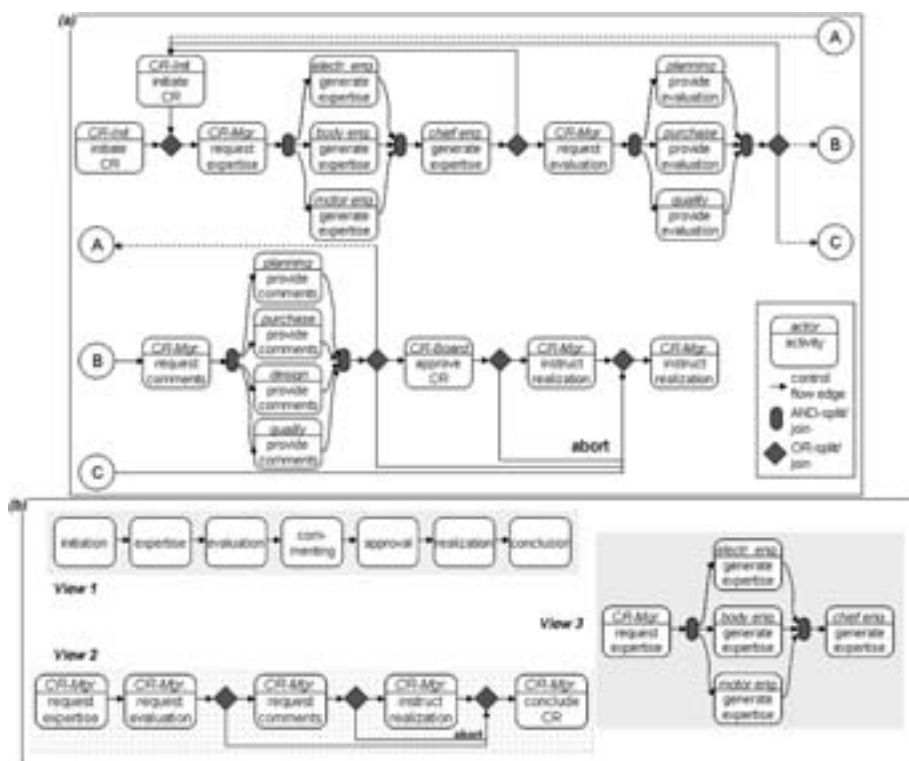


Figure 3: Automotive Domain – (a) Simplified Process of Dealing with Change Requests (CR), (b) Different Views on CR Process

Requirement 2 (Definition of static AC rights). We distinguish between “static” AC rights that are independent from the execution of a process instance, and “dynamic” AC rights for which this is not the case. The latter are based on elements such as activity status and control principles (e.g., separation of duties, dual control, and inter-case constraints) [SM02, BE01]. Regarding our CR process, a person from a specific department (e.g., mo-

tor eng.) responsible for generating expertise might not be allowed to access the *expertise document* generated by the other departments (car body eng. and electronic eng.) unless she finishes generating her own expertise. This paper focuses on static AC rights.

Requirement 3 (Usability and maintainability of AC rights). AC rights should be simple to define and easy to maintain. As discussed in [TAP05], a challenge is to balance collaboration and flexibility; i.e., we need to ensure that the advantages provided by process-aware IS are not reduced by AC rights too rigidly defined. For this purpose, abstractions are required at the objects' level. In order to specify AC rights at different levels of granularity, we need to define hierarchies on objects; e.g., it might be reasonable to authorize a manager to access all running CR process instances. However, regular users might only have access to specific CR instances (e.g., CR initiators only have the right to access CR process instances that correspond to change requests initiated by them).

Table 1 gathers major requirements identified. The ones highlighted (i.e., R1, R2, and R3) are addressed by the solution proposed in Section 5.

Table 1: Access Control Major Requirements

| Requirements | Requirements' description |
|--------------|--|
| R1 | Definition of AC rights at a fine-grained level R1.1 Meeting a spectrum of confidentiality R1.2 AC rights definable on diff. aspects of the mod./inst. levels |
| R2 | Definition of static AC rights |
| R3 | Usability and maintainability of AC rights |
| R4 | Definition of dynamic AC rights |
| R5 | Definition of AC rights on the visualization of a collection of processes |
| R6 | Definition of AC rights for the look-ahead problem |
| R7 | Completeness of the AC component |

4 Candidate Solution Approaches for Access Control

Among a list of possible AC approaches, we feature two candidate solutions that we study and compare: the view- and the object-based approach. In both approaches we follow the main idea proposed by a generalized AC approach; i.e., RBAC (Role-Based Access Control) [FSG⁺01], in which AC rights are not directly linked to concrete users, but to roles. The *view-based* approach consists of defining one basic view per user role; this view implicitly reflects the AC rights of the role over a process by only showing the information to be accessed by users with the respective role. The *object-based* approach consists of defining, for each role, AC rights on the different aspects of a process (e.g., activity, activity attributes, process instance). Section 4.1 illustrates the two featured approaches. Section 4.2 then summarizes their advantages and drawbacks. This helps us to clearly motivate the object-based approach as the one retained and elaborated in the following.

4.1 Description of Solution Approaches

View-based Approach. Considering a particular process model such as the CR process (cf. Fig. 3a), a number of views could be (manually) defined on this process. Each of them would then reflect the information accessible for users with a particular role. Access rights

over the process may be derived implicitly from each view. Suppose the following views are defined on the CR process (cf. Fig. 3b): (*View 1*) High-level view on CR process, (*View 2*) View on expertise activities of CR process, and (*View 3*) View on request activities of CR process. Then one basic view per role may be defined: (“*general manager*”, *View 1*), (“*CR manager*”, *View 2*), and (“*engineer*”, *View 3*). Each of the views implicitly reflects the read access rights of the particular role:

- A *general manager* may access high-level activities like initiation, expertise, evaluation, commenting, and so on.
- *CR managers* may access activities request expertise, request evaluation, request comments, instruct realization, and conclude CR.
- *Engineers* may access concrete activities request expertise and generate expertise.

Object-based Approach. It consists of explicitly defining an extensible set of access rights for each role:

- (“*general manager*”, {initiation, expertise, evaluation, commenting, approval, realization, conclusion}, *Read*)
- (“*CR manager*”, {request expertise, request evaluation, request comments, instruct realization, conclude CR}, *Read*)
- (“*engineer*”, {request expertise, generate expertise}, *Read*)

A view may then be generated for a specific user based on the access rights associated with the role(s) played by this user. As an example, a view such as *View 3* illustrated in Fig. 3b would be generated for motor engineer *John Smith*.

4.2 Solution Approaches: Advantages and Drawbacks

View-based Approach. The most obvious advantage comes from the fact that an existing concept (e.g., View Definition Language [Bob08]) can be explicitly reused in order to reflect the access rights over processes. Hence, there is no need for defining a new AC language assuming that the process-aware IS clearly supports a View Definition Language [Bob08]). However, three drawbacks can be identified:

Costly maintenance of views: Consider a process model *P* together with the views derived from it. Suppose a modification is brought to *P*: (1) the views affected by this change have to be identified possibly among a large number of existing views; (2) the identified views have to be adapted to reflect the change of *P*. This adaptation should be done without any failure; (3) the adapted views imply an implicit modification over AC rights.

Complexity of views combination: Since a user may play more than one role (e.g., *John Smith* being a general manager as well as a motor engineer), we must be able to combine multiple views (e.g., *View 1* and *View 3*). The resulting view, automatically generated or

manually modeled out of multiple views, will be shown to the user. On the one hand, we are facing a combinatorial problem (i.e., the different ways of arranging views in order to combine them). On the other hand, conflicts may exist between access rights reflected by the views to be combined. Such conflicts, first, must be detected, and second, be solved, probably by applying specific conflict resolution policies [dVSJ05, JSSS01].

Occurrence of redundant information due to lack of abstraction: Suppose that a specific role R has access, among other things, to a specific activity A in all processes involving A . Using the view-based approach, this access right would be reflected by showing A within all the views respectively defined on the processes containing A . This leads to redundant information due to the definition of access rights at the level of process models, not involving functional models (cf. Sect. 2). The redundancy of information is an issue not only for the view-based approach, but for other approaches as well, as long as the notion of abstraction is missing (e.g., at the level of activities). However, redundancy has more impact in conjunction with the view-based approach than in conjunction with the object-based one.

Object-based Approach. The main advantage of this approach is threefold. Indeed, the drawbacks identified for the view-based approach appear to be advantages here. First, there is no maintenance of views; the cost behind the maintenance operation is abolished. Second, views have not to be combined and hence the complexity behind this operation does not exist. Third, if it is possible to define different levels of abstractions on objects, this will reduce redundancy when specifying access rights. The object-based approach may be criticized for not being intuitive since AC rights, instead of basic views, are initially defined for each role. However when compared with the drawbacks of the view-based approach, we voluntarily accept this only criticism, and select the object-based approach in order to elaborate the core solution for our logical AC model.

Table 2 summarizes the most important criteria that play either in favor of or against each of the considered approaches. As we can see, among five criteria, three play in favor of the object-based approach, while only one criterion plays in favor of the view-based approach.

| Table 2: Comparison of the View-based and Object-based Approaches | | |
|---|------------|--------------|
| Criteria/Approaches | View-based | Object-based |
| Ease of AC rights definition | + | - |
| Ease of AC rights maintenance | - | + |
| Ease of conflicts resolution | - | - |
| Ease of AC rights combination | - | + |
| Redundancy-free | - | + |
| + Criterion plays in favor of the approach | | |
| - Criterion plays against the approach | | |
| * This criterion is reduced to the "Ease of conflicts resolution" criterion | | |

5 An Access Control Model

An AC model for process monitoring must allow to restrict access to authorized users only. Sect. 5.1 presents our formal framework for defining and manipulating AC rights. Sect. 5.2 and Sect. 5.3 discuss AC model extensions for coping with the problem of users playing multiple roles, and for addressing usability and maintainability issues.

5.1 Core AC Model

The specification of an AC module at the process monitoring level requires, first and foremost, the definition of access rights. A first step towards meeting *Req. R1* (cf. Table 1) consists of defining access rights on *attributes* associated with specific process aspects that we call *objects*. Activities, process models or process instances are examples of accessed objects; attributes, indeed, reflect fine-grained characteristics of such objects. For this purpose, we first formally define the link between an object and its associated attributes.

Definition 1 (Set of Attributes Associated with an Object) *Let $ObjSet$ and $AttSet$ respectively be the set of objects and the set of attributes involved in the process monitoring component. Then function $attributeSet$ determines all attributes associated with an object $obj \in ObjSet$. Formally: $attributeSet: ObjSet \mapsto AttSet^P$ with $\forall att \in attributeSet(obj): att$ is a valid attribute defined on obj .*

We associate with every object involved in the process monitoring component a set of attributes. Formally: $\forall obj \in ObjSet: attributeSet(obj) \subseteq AttSet$

In order to illustrate Def. 1, we reconsider the process from Fig. 3a. For the sake of simplicity, we only retain the concrete concept of *activity* instead of the generalized one of *object*. Let $ObjSet = \{\text{request expertise, generate expertise, request evaluation, provide evaluation, request comments, provide comments}\}$ be a set of activities involved in the CR process. Let further $AttSet = \{Att_1, Att_2, Att_3, Att_4, Att_5\}$ be the set of attributes involved in the CR process. Taking into account Def. 1, suppose that the set of attributes associated with each activity is captured as follows: $attributeSet(\text{req. expertise}) = \{Att_1, Att_3\}$; $attributeSet(\text{gen. expertise}) = \{Att_1, Att_2, Att_4, Att_5\}$; $attributeSet(\text{req. evaluation}) = \{Att_1, Att_3\}$; $attributeSet(\text{prov. evaluation}) = \{Att_1, Att_2, Att_5\}$; $attributeSet(\text{req. comments}) = \{Att_1, Att_3\}$; $attributeSet(\text{prov. comments}) = \{Att_1, Att_2\}$. We may think of Att_1 as the *activity status* that could take values from the set $\{\text{NotActivated, Activated, Running, Completed, Skipped}\}$. Att_2 may be the *starting date/time* of an activity. Att_3 could be the *employee black list* with possible values $\{\text{Yes, No}\}$ specifying whether this list should be taken into account (or not) when employees are chosen to work on a specific task (e.g., generate expertise). If this list is taken into account, employees on black list may be excluded from those that may work on the task.

Based on Def. 1, we retain two types of information that may be checked/read: *the existence* and *the value* of an object's attribute. We distinguish between two different spectra of confidentiality defined on this information: 1) "Allow"/"don't allow" to check existence of an attribute within an object; 2) "Allow"/"don't allow" to read the value of an attribute within an object, or allow to read another form of the value. From this we derive Def. 2.

Definition 2 (Access Control on Existence/Value of Attribute) *Let (obj, att) ($obj \in ObjSet, att \in attributeSet(obj)$) denote an attribute att associated with object obj . Then $Exist_{obj,att}$ determines whether it is allowed for someone (or not) to check the existence of attribute att within object obj ; $Val_{obj,att}$ determines whether it is allowed for someone*

(or not) to read the value of attribute *att* within object *obj*. Formally:

$$Exist_{obj,att} := \begin{cases} 0 & \text{if not allowed to check existence of } att \text{ within } obj \\ 1 & \text{if allowed to check existence of } att \text{ within } obj \end{cases}$$

$$Val_{obj,att} := \begin{cases} 0 & \text{if not allowed to read value of } att \text{ within } obj \\ 1 & \text{if allowed to read only another form of value} \\ 2 & \text{if allowed to read value of } att \text{ within } obj \end{cases}$$

Back to our example from Fig. 3a, suppose role “engineer” has the following access rights on the CR process: access to activities *request expertise* and *generate expertise*, access to the value of *Att₁* and to another form of the value of *Att₂*, and access to the existence of *Att₃* within *request expertise*. Taking into account Def. 2, the AC on the existence/value of the different attributes can be captured as follows:

$$Val_{generate\ expertise, Att_1} = 2, Val_{generate\ expertise, Att_2} = 1, \\ Val_{request\ expertise, Att_1} = 2, Exist_{request\ expertise, Att_3} = 1$$

By default, we may suppose that the *closed policy*, considered as a classical approach for AC [Cas95], applies. If not specified otherwise:

$$Val_{obj,att} = 0 \text{ and } Exist_{obj,att} = 0, \forall obj \in ObjSet, att \in attributeSet(obj)$$

In this context, two classical approaches for AC are discussed in literature [Cas95]: *closed policy* where positive rights need to be specified explicitly, and *open policy* where negative rights need to be specified explicitly. The closed policy approach is known to ensure better protection than open policy. In the latter, the need for protection is not strong: by default, access is to be granted. Intuitively, we may also suppose that a specific operation prevails on another (cf. Fig. 4); e.g., whenever it is allowed to read the value of an attribute, this implies that it is also allowed to read another form of the value, and to check the existence of the attribute. Note that positive rights prevail on negative ones, i.e., positive rights are on bottom of the scale in Fig. 4. This is because of the closed policy adopted. Taking into account this scale, the following set of access rights is retained:

$$Val_{generate\ expertise, Att_1} = 2, Val_{generate\ expertise, Att_2} = 1, \\ Val_{request\ expertise, Att_1} = 2, Exist_{request\ expertise, Att_3} = 1, \\ Exist_{generate\ expertise, Att_4} = 0, Exist_{generate\ expertise, Att_5} = 0, \\ Exist_{Activity, Attribute} = 0, \forall Activity \in ObjSet \setminus \{request\ expertise, \\ generate\ expertise\}, Attribute \in attributeSet(Activity)$$

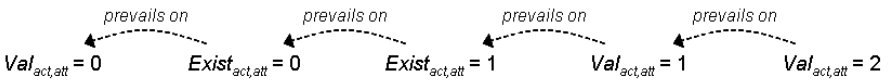


Figure 4: Prevalence of Access Rights

AC rights being clearly defined, we present now a mechanism consisting of two functions that respectively return 1) whether or not an attribute is associated with an object, 2) the exact value or an abstraction of the value of an attribute.

Definition 3 (Existence/Value of Attribute) *Let (obj, att) ($obj \in ObjSet, att \in attributeSet(obj)$) be an attribute associated with an object. Let Val be a function on $ObjSet \times AttSet$, $Val: ObjSet \times AttSet \mapsto Dom_{AttSet} \cup \{Undefined\}$. Val reflects for each $(obj, att) \in ObjSet \times AttSet$ its current value from domain Dom_{AttSet} or the value “Undefined” if att has not been written yet. Let $FunctionSet$ be the set of functions that can be applied on the value of an attribute in order to provide another form of this value. For defining the specific function that can be applied on a specific attribute, we need the function:*

$fa: ObjSet \times AttSet \mapsto FunctionSet \cup \{Undefined\}$ which maps each couple $(obj, att) \in ObjSet \times AttSet$ to a specific function from $FunctionSet$ or to “Undefined” if $att \notin attributeSet(obj)$ or no function is defined.

Then, f returns either the name of attribute att within object obj , or “Undefined”; h determines either the value or another form of the value of attribute att within object obj , or “Undefined”. Formally:

$$f: ObjSet \times AttSet \mapsto AttSet \cup \{Undefined\}$$

$$with f(obj, att) := \begin{cases} att & \text{if } Exist_{obj,att} = 1 \wedge att \in attributeSet(obj) \\ Undefined & \text{otherwise} \end{cases}$$

$$h: ObjSet \times AttSet \mapsto Dom_{AttSet} \cup Dom_{FunctionSet} \cup \{Undefined\}$$

$$with h(obj, att) := \begin{cases} Undefined & \text{if } Val_{obj,att} = 0 \\ fa(obj, att)(Val(obj, att)) & \text{if } Val_{obj,att} = 1 \\ Val(obj, att) & \text{if } Val_{obj,att} = 2 \end{cases}$$

$$Dom_{AttSet} = \bigcup_{att \in AttSet} Dom_{att}$$

$$Dom_{FunctionSet} = \bigcup_{fct \in FunctionSet} Dom_{fct}$$

If we go back to our example, applying Def. 3 would lead to the following existence/value of the different attributes:

$$h(generate\ expertise, Att_1) = Val(generate\ expertise, Att_1)$$

$$f(generate\ expertise, Att_1) = Att_1$$

$$h(generate\ expertise, Att_2) = fa(generate\ expertise, Att_2) \\ (Val(generate\ expertise, Att_2))$$

$$f(generate\ expertise, Att_2) = Att_2$$

$$h(request\ expertise, Att_1) = Val(request\ expertise, Att_1)$$

$$f(request\ expertise, Att_1) = Att_1$$

$$h(request\ expertise, Att_3) = Undefined$$

$f(\text{request expertise}, \text{Att}_3) = \text{Att}_3$
 $h(\text{Activity}, \text{Attribute}) = f(\text{Activity}, \text{Attribute}) = \text{Undefined}$
for all other combinations of activities and attributes

The result of applying Def. 3 on our CR process, taking into account specific access rights assigned to role “engineer”, is illustrated in Fig. 5.

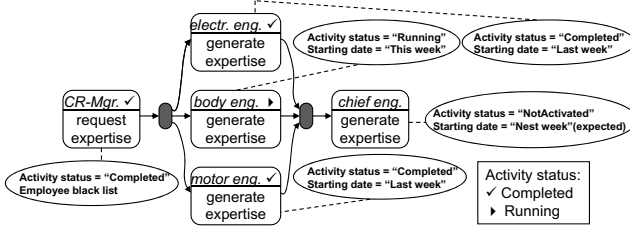


Figure 5: View on CR Process Provided to Role “Engineer”

5.2 Extended AC Model - Users Playing Multiple Roles

In this section, we recognize and point out the fact that a user may play more than one role leading to inconsistencies between the AC rights associated with each of the different roles. As example, a user may play roles “manager” and “engineer”. On the one hand, engineers may not be given access to private information. On the other hand, managers may need to access private documents, and access to such information may be given to them. In this context, a number of conflict resolution policies are discussed in literature [dVSJ05, JSS01, FGS94, SD92]. None of them represents “the perfect solution”. Whichever policy we take, we will always find one situation for which it does not fit. [dVSJ05] states some problems of the different policies in conjunction with specific scenarios. Interestingly, conflicts may result either from explicitly defining negative AC rights, or from applying the closed policy. In the latter case, a simple solution approach may be to neglect negative AC rights derived from the used policy. Conflict resolution policies should be applied in the former case. For lack of space, we abstain from discussing this matter here.

5.3 Extended AC Model - Compact Definition of AC rights

So far, we have expressed that a certain attribute is allowed to be accessed (or not) within a certain object, particularly a certain activity. However, we must also be able to state within which processes this is allowed, i.e., what is the *context* of the AC to be defined. Candidates for the context are the entire process monitoring component (All), a group of process models, a particular process model, a group of process instances related to a particular process model, and a process instance.

The example elaborated in Section 5.1 presents a set of AC rights defined on a spe-

cific process model: CR_M . We may think of the following representation: $(CR_M, Val_{generate\ expertise, Att_1} = 2)$ stating that the value of Att_1 from activity `generate expertise` is allowed to be read within process model CR_M . Suppose that AC rights are defined on a set of process models (e.g., M_1, M_2, M_3). This would lead to a set of couples: $(M_1, Val_{generate\ expertise, Att_1} = 2)$, $(M_2, Val_{generate\ expertise, Att_1} = 2)$, $(M_3, Val_{generate\ expertise, Att_1} = 2)$. Hence, we recognize the need for abstraction at the objects' level in order to compact the definition of AC rights reducing redundancy as much as possible. Therefore, one feasible way is to organize objects hierarchically (cf. Fig. 6): "All" at the top level, "Group of process models" at the next level down, "Process model" at the level just after, etc., and to propagate AC rights top-down. This allows us to meet the AC rights usability and maintainability requirement (cf. R3 in Table 1). Going back to our example, a group of process models $G_M = \{M_1, M_2, M_3\}$ would be defined, and the set of three couples would be reduced to the following couple: $(G_M, Val_{generate\ expertise, Att_1} = 2)$. This approach would also simplify the definition of exceptions; e.g., it would be easy to express that no restrictions exist at all regarding accesses within any of the defined processes except the following: no accesses are allowed to activity `approve CR` within the CR process model. This would be reduced to: $(All, Val_{All, All} = 2)$ (i.e., access is given to everything in order to bypass the closed policy), and $(CR_M, Exist_{approve\ CR, All} = 0)$ (i.e., access is retrieved from `approve CR` within CR_M).

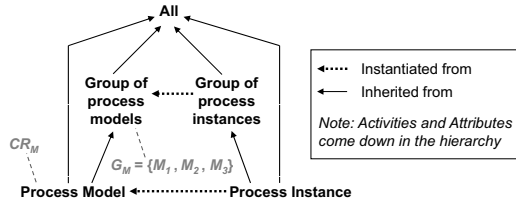


Figure 6: Objects' Hierarchy

6 Related Work

The provision of adequate security mechanisms is indispensable for any IS. Particularly, in the context of process-aware IS such as ADEPT [RDB03], approaches have been proposed for dealing in a secure way with specific issues related to process management. As an example, Weber et al. propose an extension to RBAC in order to support process changes safely [WRWR05]. In the CEOSIS project, Rinderle and Reichert address changes that may occur within organizational structures [RMR07]. They discuss how to support such changes, and how to adapt access rules when the underlying organizational model is changed [RMR08, RMR09]. However, to our best knowledge, no research work has yet addressed the problem of fine-grained AC in conjunction with process data integration and process monitoring [Mue01, JHBK04]. This also applies in respect to existing process performance management tools (e.g., ARIS PPM).

Some of the aspects retained in this paper have already been introduced by others. The fine-grained control was discussed in [TAP05] as one of the collaborative environment

factors that determine the usability of a specific AC model. The authors argue that it is not sufficient to define AC rules only for groups of users on clusters of objects. A user might need a specific permission on an instance of an object at a particular point (i.e., time) in the collaboration session. In our approach, we were more explicit when defining AC rights at a fine-grained level: 1) we introduced the *spectrum of confidentiality* concept that would reflect the “specific” permission to grant or to revoke, and 2) we hierarchized objects such that AC rights may be defined in a *compact way* on the *different aspects* of the process model and instances. In [TAP05], no details are given regarding *time* (i.e., a permission is valid only for a specific time space). This is an interesting point to be further investigated. In the context of adaptive process-aware IS, Weber et al. propose the definition of process type dependent AC rights [WRWR05]. Only change commands that are useful within a particular *context* are allowed. This idea can be compared to our approach of specifying the context of an AC right. However, both approaches focus on different aims. [WRWR05] provides assistance for users when performing a change, whereas in this paper, the context notion is used for defining AC rights in a more focused way. We refer to [KR09, RMR09] for discussions of other AC frameworks in process-aware IS.

7 Summary and Outlook

We identified AC requirements in the context of process monitoring. We then presented possible solution approaches for major requirements, and we motivated the objects-based approach that we used for proposing a core AC model for process monitoring. Two extensions to this model were also discussed: the first one deals with the problems that may appear when a single user plays more than one role; the second extension introduces the “context” notion and discusses the compact definition of AC rights taking into account a defined objects’ hierarchy. Major requirements were addressed using the proposed AC model and its extensions. In future work, we will address requirements R4-7 (cf. Table 1). Our research work will also include the investigation of advanced issues such as the aggregation and the definition of AC rights on data elements and other process aspects.

References

- [BBR06] R. Bobrik, T. Bauer, and M. Reichert. Proviado Personalized and Configurable Visualizations of Business Processes. In *Proc. EC-WEB’06*, LNCS 4082, pages 61–71, 2006.
- [BE01] R.A. Botha and J.H.P. Eloff. Separation of Duties for Access Control Enforcement in Workflow Environments. *IBM Systems Journal*, 40(3):666–682, 2001.
- [Bob08] R. Bobrik. *Konfigurierbare Visualisierung komplexer Prozessmodelle*. PhD thesis, University of Ulm, 2008.
- [BRB05] R. Bobrik, M. Reichert, and T. Bauer. Requirements for the Visualization of System-Spanning Business Processes. In *Proc. DEXA’05 Workshops*, pages 948–954, Copenhagen, August 2005.

- [BRB07] R. Bobrik, M. Reichert, and T. Bauer. View-Based Process Visualization. In *Proc. BPM'07*, LNCS 4714, pages 88–95, 2007.
- [Cas95] S. Castano et al. *Database Security*. Addison Wesley, 1995.
- [dVSJ05] S. De Capitani di Vimercati, P. Samarati, and S. Jajodia. Policies, Models, and Languages for Access Control. In *Proc. Int'l Workshop DNIS'05*, pages 225–237, Aizu-Wakamatsu, March 2005.
- [FGS94] E.B. Fernandez, E. Gudes, and H. Song. A Model for Evaluation and Administration of Security in Object-Oriented Databases. *IEEE ToKDE*, 6(2):275–292, 1994.
- [FSG⁺01] D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, and R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM ToISS*, 4(3):224–274, 2001.
- [JHBK04] S. Junginger, H. Huehn, F. Bayer, and D. Karagiannis. Workflow-based Business Monitoring. In *Workflow Handbook 2004*. Future Strategies, Lighthouse Point, 2004.
- [JSSS01] S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian. Flexible Support for Multiple Access Control Policies. *ACM ToDS*, 26(2):214–260, 2001.
- [KR09] V. Künzle and M. Reichert. Integrating Users in Object-aware Process Management Systems: Issues and Challenges. In *Proc. Business Process Management Workshops 2009*. Springer, 2009.
- [Mue01] M. zur Muehlen. Workflow-based Process Controlling. In *Workflow Handbook 2001*. Future Strategies, Lighthouse Point, 2001.
- [RDB03] M. Reichert, P. Dadam, and T. Bauer. Dealing with Forward and Backward Jumps in Workflow Management Systems. *Software and Systems Modeling*, 2(1):37–58, 2003.
- [RMR07] S. Rinderle-Ma and M. Reichert. A Formal Framework for Adaptive Access Control Models. In *Journal of Data Semantics, IX*, LNCS 4601, pages 82–112, 2007.
- [RMR08] S. Rinderle-Ma and M. Reichert. Managing the Life Cycle of Access Rules in CEOSIS. In *Proc. EDOC'09*, pages 257–266, 2008.
- [RMR09] S. Rinderle-Ma and M. Reichert. Comprehensive Life Cycle Support for Access Rules in Information Systems: The CEOSIS Project. *Enterprise Information Systems*, 3(3):219–251, 2009.
- [SD92] H. Shen and P. Dewan. Access Control for Collaborative Environments. In *Proc. CSCW'92*, pages 51–58, November 1992.
- [SM02] A. Schaad and J. Moffett. A Framework for Organisational Control Principles. In *Proc. ACSAC'02*, pages 229–238, Las Vegas, December 2002.
- [TAP05] W. Tolone, G.-J. Ahn, and T. Pai. Access Control in Collaborative Systems. *ACM Computing Surveys*, 37(1):29–41, 2005.
- [Wes07] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.
- [WRWR05] B. Weber, M. Reichert, W. Wild, and S. Rinderle. Balancing Flexibility and Security in Adaptive Process Management Systems. In *CoopIS'05*, LNCS 3760, pages 59–76, 2005.

A Survival Analysis of Application Life Spans based on Enterprise Architecture Models

Stephan Aier¹, Sabine Buckl², Ulrik Franke³, Bettina Gleichauf¹,
Pontus Johnson³, Per Närman³, Christian M. Schweda², Johan Ullberg³

¹Institute of Information Management
University of St. Gallen
Müller-Friedberg-Str. 8
CH-9000 St. Gallen
{stephan.aier,bettina.gleichauf}@unisg.ch

²Lehrstuhl für Informatik 19 (sebis)
Technische Universität München
Boltzmannstr.3
D-85748 Garching
{buckls,schweda}@in.tum.de

³Industrial Information and Control Systems
Royal Institute of Technology (KTH)
Osqudas väg 12
SE-100 44 Stockholm, Sweden
{ulrikf,pj101,pern,johanu}@ics.kth.se

Abstract: Modern enterprises face the challenge to survive in an ever changing environment. One commonly accepted means to address this challenge and further enhance survivability is enterprise architecture (EA) management, which provides a holistic model-based approach to business/IT alignment. Thereby, the decisions taken in the context of EA management are based on accurate documentation of IT systems and business processes. The maintenance of such documentation causes high investments for enterprises, especially in the absence of information on the change rates of different systems and processes. In this paper we propose a method for gathering and analyzing such information. The method is used to analyze the life spans of the application portfolio of three companies from different industry sectors. Based on the results of the three case studies implications and limitations of the method are discussed.

1 Introduction

The rate of change in the economic environment of enterprises has increased over the past few years [RWR06; Wa05]. Some underlying factors are increased customization of products and services coupled with globalization and a more competitive market situation. Furthermore, regulations like the Sarbanes Oxley Act, Basel II, etc. need to be

met [La05]. Enterprises have to continuously adapt to these environmental changes, by aligning their business, applications, data, and infrastructure to the new requirements. One commonly accepted means to guide such adaptations is enterprise architecture (EA) management, which is a holistic model-based approach to enterprise engineering, specifically addressing business/IT alignment.

The ability to make informed decisions in the complex and highly interdependent area of EA management is closely linked to accurate descriptions and documentations of the EA. An EA model must be both up-to-date [Ci01] and appropriate with respect to the decisions it is to support [JE07; La05]. Planning based on obsolete data or decisions made with insufficient information will almost certainly have an unfavorable influence on the EA, and thus in the long run have a negative business impact.

Due to the continuous changes going on in different parts of any company's structure, EA descriptions will inevitably become obsolete at some point. Therefore, it seems reasonable to establish a maintenance process for EA models, continuously updating the EA documentation. However, such a process is not for free. EA descriptions are extensive, and the cost of collecting information and creating architecture models is considerable. To make this cost-benefit trade-off, enterprises would benefit from a model describing how fast certain parts of the enterprise's structure are likely to change. Based on such a model, an organization could decide not to update the descriptions of selected parts, as their frequency of change is so low that no relevant gain is expected from frequent updating. In a manner of speaking, this amounts to find an optimal EA *sampling rate*.

In this article, we approach the topic of change frequencies for applications as one of the main enterprise artifacts related to business/IT alignment [AW09] with a case study based research paradigm. While change in an EA may include modifications, introduction, and removal of individual elements, we only consider the latter two phenomena in this context. Thus, mere modifications of applications are not within the scope of this article. Rather, the research questions posed are:

- How can life spans of different enterprise artifacts be assessed?
- What are the approximate life spans of such enterprise artifacts?

The rest of the article unfolds as follows. Related work is presented in section 2. There, we discuss the relevant literature on EA model maintenance and prepare our analysis of life spans by giving an overview of appropriate models found in the literature. Based on these discussions, section 3 proposes an analysis model for the decay of artifacts found in EA models. Subsequently, section 4 presents the three case studies, the analysis results of which are presented in section 5. Section 6 contains a concluding discussion on the empirical results, and gives an outlook to further areas of research.

2 Related work

This section elaborates on related work in the field of EA model maintenance (section 2.1) and from the field of methods and models for analyzing decay (section 2.2).

2.1 Enterprise architecture model maintenance

A multitude of methods for EA management has been developed by researchers and practitioners (e. g. [Az05; Az06; BK05; Dv01; Og09; SH93; Wa05]). These methods usually distinguish 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, (e) EA implementation, and (f) EA analysis based on EA models.

None of these approaches to EA management pays much attention to specifying maintenance procedures for the EA model in detail. While [Ci01; If99] and [Wa05] mention an EA maintenance process, the corresponding activities are not specified in detail, and neither specific roles nor responsibilities are defined. TOGAF [Og09] introduces the objective of *ensuring that the baseline architectures continue to be fit-for-purpose* as an important aspect of the phase *Architecture Change Management*. However, no details on how to accomplish this objective are given by the framework.

In the academic research community, maintenance procedures for EA models have recently gained increased interest. In an initial paper, [FAW07] developed a systematic decentralized EA maintenance approach describing maintenance processes and roles. [Mo09] present six EA management process patterns related to EA maintenance. Several of these patterns address lifecycles and dynamics of EAs. However, neither [FAW07] nor [Mo09] describe schedules for maintaining EA models. To conclude, requirements on EA maintenance are abundant in the literature, but the concrete solutions are very scarce.

2.2 Methods and models for analyzing decay

Quantitative theories of aging, mortality, and life span have a long history in medicine, demographics, and insurance mathematics. As far as biological systems are concerned, *survival analysis* is used to model the effect of death. Similar analyses can be performed on non-biological systems, e.g. mechanical ones, in the field of reliability theory. This theory aims to analyze, estimate, and predict life span distributions of systems and their components [BPH65; GG01]. Typically, a *reliability function* $S(t)$ is defined as the probability that a system carries out its mission through a time t . Complementing the reliability function, the *hazard function* $\lambda(t)$ is defined as the probability that a failure occurs at a specific time under the condition that it has not occurred up to that point in time.

One of the most popular laws regarding the hazard function was found by Benjamin Gompertz, today known as *Gompertz Law of Mortality* [Go25]. He analyzed the mortality of adult humans in respect to their age, and found that the corresponding hazard func-

tion increases in geometrical progression with the age. Based on this observation, he proposed a mathematical model to explain this behavior – the *Gompertz distribution*.

A special application of survival functions is presented by [FCH83]. The authors investigate the death rates of organizations, by eliciting survival functions and hazard functions for organizations. They adjust the *Makeham extension* of the Gompertz Mortality Law, based on the assumption that the death rate of organizations decreases with age (liability of newness) and taking into account influence factors like size of the organization or environmental changes. By the means of extensive explorative analyses of national labor unions, local newspaper organizations, and semiconductor manufacturing firms, they calculate estimates of organizational death rates.

The Gompertz Mortality Law and its extensions are prominently applied for biological and sociological system. In more technical application areas, failure and reliability analyses give rise to different kinds of laws. For instance, the *Weibull distribution* can be used to model a variety of failure behaviors [Ca03]. The use of the Weibull distribution as well as the Gompertz distribution requires a long period of observation and a complete data set in order to determine the estimators of the distribution's parameters. However, as such data sets are rare in the field of EA, a more flexible method is needed. In particular, information on the age of an EA artifact such as an application is very hard to collect. Most information available is on end-of-life events for EA but little on their creation. Furthermore, the *center of attention* of EA modeling undergoes shifts during the years of observation such that new EA artifacts appear during observation or are lost.

This situation is very similar to some encountered within medical statistics [BI00]: In medical statistics, some patients are lost from observation while others are added as the study progresses. Little information is known regarding the start date of e.g. the cancer, but more is known about the end date. For these reasons we will employ methods from medical statistics for our life-span analysis. The next section elaborates on this.

3 Survival data analysis with the life table method

Data constrained in the abovementioned way can be analyzed using the *life table method* [BI00], which provides the analyst with a tool with which it is possible to keep track of the probabilities of death and survival of the study subjects for each passing year. To summarize, the life table method does not require knowing actual start dates, allows for losing track of some observed elements and is able to derive knowledge from the observation of elements also before their terminal event. To illustrate the life table method, a generic table is shown below in Table 1, along with a description of its columns.

The terminal event, be it death from cancer or an application taken out of service, is called the *endpoint* and is here referred to as the *death* of the subject. Generally there is also a wish to evaluate the survival well before all the endpoints are known, i.e. while a good fraction of patients or applications are still alive or in use. Data that is only known to be greater than some value is called *right censored* (or just *censored*, for short). This is the kind of data obtained when examining patients or applications that recently entered

the study and have only been observed for a short time and have yet to reach their end-points. These subjects are called *withdrawn*.

Table 1: Generic life table [BI00].

| Period (years, etc.) | Number at start | Withdrawn during period | At risk | Deaths | Prob. of death | Prob. of sur- viving period x p_x | Cumulative prob. of surviving x pe- riods P_x |
|-------------------------|--------------------|----------------------------|-----------------|----------|-------------------|---|--|
| x | n_x | w_x | r_x | d_x | q_x | | P_x |
| 1 | n_1 | w_1 | $n_1 - 1/2 w_1$ | d_1 | d_1/r_1 | $1 - d_1/r_1$ | p_1 |
| \vdots | \vdots | \vdots | \vdots | \vdots | \vdots | \vdots | \vdots |
| T | n_T | w_T | $n_T - 1/2 w_T$ | d_T | d_T/r_T | $1 - d_T/r_T$ | $p_T P_{T-1}$ |

Table 1 depicts important life table concepts. A study period, x , of $[1, T]$ periods is considered, where T corresponds to the longest time period for which there is data available, for instance 9 years if the study covers information between 2000 and 2009. The time period states nothing about the fixed point in time when the observation of a subject commenced, as in for instance the year 2004. It merely describes the amount of time a subject is observed, as in for instance four years. The number at start n_x is the number of elements in the study at the start of *their* corresponding period x under observation. n_1 is the total number of participants in the study and n_x is calculated by taking n_{x-1} and deducting the number of deaths and withdrawals (see below) during period $x-1$.

New subjects that appear after the start date of the survey may also be included in the data analysis. For instance, an application taken into service in year 2006 may be included although the oldest data is from 2000. These new subjects enter the study at time period 1 and can only contribute to the statistics for a limited period of time, since data about their future survival is unavailable. At the relative point in time that a subject becomes unobservable it is declared withdrawn. In the example above applications taken into service 2006 can contribute with survival data for three years until today's date (2009) is reached and thus they are declared withdrawn at the fourth year. w_x represents the amount of withdrawn elements. When reading our statistics it is essential to bear in mind that period 1 is the first period after an application is observed; it is *not* a particular year such as 2001. To summarize; the concept of withdrawn elements is used to reflect the fact that a subject that only can be observed for e.g. four years provides survival information during these years but tells us nothing about the probability of surviving the fifth year.

To get an appropriate number of elements at risk, i.e. the population in which deaths can occur, the number at the start of the period is reduced by half the number of withdrawn elements during the period. This corresponds to saying that the withdrawn cases, on average, contribute half a period of risk to the population. The number of deaths d_x during period x is an observed amount. We consider artifacts that are listed in an EA repository

to be *alive*, i.e. in use. Artifacts not listed in a repository at a given point in time of consideration are declared *dead*, i.e. taken out of service.

The probability of death q_x in period x is simply calculated as the fraction of those at risk that were actually observed to die. The probability p_x of surviving is the complement of q_x . Finally, the cumulative probability P_x of surviving x periods is calculated as the product of the probabilities of surviving the periods. For the first period, $P_1 = p_1$ holds. For the second period, P_2 is the probability of surviving up to the start of period 2, i.e. P_1 multiplied by the probability of surviving period two as well, i.e. p_2 . The same reasoning holds in the general case, so $P_t = P_{t-1} p_t$ for all $t > 1$.

A natural use of the data in a life table is to draw a graph of the cumulative survival probability, usually in a step-wise fashion as to underline the inexactness of the estimation. This is called a *survival curve*. In section 4 we will provide life tables for each case study. The respective survival curves are presented and discussed in section 5.

4 Case studies

Our findings are based on three case studies from different industry sectors, presented in the following. Each case starts with a characterization of the enterprise at hand, including its established EA management approach. Subsequently, characterizations of the used data sets are given, potential errors are discussed, and analysis results are presented.

The data sets used originate from different sources, e.g. application catalogs, MS Excel files, and exports from specialized EA management repositories. As the EA management initiatives analyzed in the three case studies have very disparate backgrounds, including using different tools to store the data, the different data sets of case studies B and C are drawn from a single source, while the data sets of case study A originate from diverse. In order to compare the data sets, a mapping of the different schemas used to document the data about applications had to be performed within each case. During the comparison, two records were regarded to refer to the same application either if they had the same *id* or if they possessed the same *name*. This id or name matching represents a potential source of errors in the data set. To detect and correct such errors, occasional reviews with the respective stakeholders at the industry partners were performed during the conduction of the studies.

4.1 Company A

Company A is one of the principal energy companies in Europe and among the five largest generators of electricity. After the deregulation of several European energy markets in the nineties, company A expanded through acquisitions and has become a big actor in several European countries. As a consequence, the company's IT portfolio has become rather heterogeneous and in the past couple of years there have been several activities to consolidate the application landscape.

Since the generation, transmission, and distribution of electricity and heat are geographically bound activities, company A is organized into a number of geographical business groups each of which is functionally divided into business units according to their contribution to the electricity or heat value chain. The IT infrastructure is centralized in a shared services company but the responsibility for the processes and the applications are distributed amongst the business groups and the business units. Each business group has a coordinating CIO function responsible for the EA work. The group CIO function in turn coordinates and supports local and global EA initiatives within the company.

The empirical data presented here originates from three sources. The first source is the application catalogue that was created in 2000 in order to address the perceived Y2K threat. Secondly, another application catalogue was established in late 2005 and the beginning of 2006 as a part of the first embryo of a group-wide EA program. Finally, the third source is a group-wide application catalogue compiled in 2009, which is used to identify consolidation opportunities within the company.

The coverage of the Y2K catalogue is limited to one business group and encompasses 124 critical technical systems pertaining to different business units. The second application catalogue is more detailed covering over 500 applications. The third application catalogue has an even finer granularity and includes information on which processes are supported as well as information on the age of the applications and their principal integration points. Since the different sources aren't equidistant in time the life table (which contains relative time periods) has four rows while each dataset is however only applicable to at most three of the rows.

Table 2: Life table for applications of Company A

| Time period (Year) (x) | Number at start (n_x) | Withdrawn during year (w_x) | At risk (r_x) | Deaths (d_x) | Prob. of death (q_x) | Prob. of surviving period x (p_x) | Cum. prob. of surviving the period x (P_x) |
|----------------------------|---------------------------|---------------------------------|-------------------|------------------|--------------------------|---|--|
| 1 | 577 | 0 | 577.0 | 0 | 0.00 | 1.00 | 1.00 |
| 4 | 577 | 0 | 577.0 | 235 | 0.41 | 0.59 | 0.59 |
| 7 | 342 | 250 | 217.0 | 32 | 0.15 | 0.85 | 0.51 |
| 10 | 60 | 0 | 60.0 | 15 | 0.25 | 0.75 | 0.38 |

Although the material is bound to contain some errors in minute details, the overall indications from the company is that the data is sufficient to draw conclusions concerning whether applications exist or not at the three points in time. The resulting life table is shown in Table 2 and as seen in the table the cumulative probability of an application to survive 10 years is 38%. A note to the table is that since the granularity of data is three year intervals the rows of the table should be considered to be periods rather than the individual years.

4.2 Company B

The second case is taken from the financial industry. The enterprise under consideration is an internationally operating bank from Germany. The topic of EA management has a long history in this enterprise since a merger in the year 1996. Prior to the merger both companies independently conducted enterprise-wide data modeling endeavors. After the merger, the enterprise-wide data models were maintained, although a change in the focus as well as the reach took place. In certain parts of the enterprise the focus shifted towards a strongly business process centric modeling, while other parts continued to do pure data modeling. In the year 2002 the term EA management makes its first appearance, when a project was launched to increase the business/IT alignment based on a holistic model of the relevant aspects ranging from strategy to infrastructure. In this model architectural information from different parts of the enterprise was consolidated and used to identify fields for action. The first data set analyzed originates from the aforementioned early EA management project. In order to assess the advances made in this field, a similar project was launched in the year 2005. The take-over by an international banking company at the end of 2005 changed the overall make-up of the company significantly. In particular, the IT departments of the formerly independent enterprises, as well as the IT assets developed, operated, and managed by them, were to undergo extensive changes leading to an increased centralization of structures. In 2008, this centralization process has proceeded quite far, such that the final data set from the end of the year 2008 shows the face of the EA according to the new paradigm.

The different data sets analyzed for this case study are several snapshots of the application portfolio of company B. The data sets analyzed cover about 90% of the application portfolio of company B including legacy systems of the company as well as small interim solutions. Table 3 shows the resulting life table for applications of company B, indicating that the cumulative probability to survive 6 years is 49%.

Table 3: Life table for applications of company B

| Time period (Year) (x) | Number at start (n_x) | Withdrawn during year (w_x) | At risk (r_x) | Deaths (d_x) | Prob. of death (q_x) | Prob. of surviving period x (p_x) | Cum. prob. of surviving the period x (P_x) |
|----------------------------|---------------------------|---------------------------------|-------------------|------------------|--------------------------|---|--|
| 1 | 1142 | 16 | 1134.0 | 212 | 0.19 | 0.81 | 0.81 |
| 2 | 914 | 31 | 898.5 | 109 | 0.12 | 0.88 | 0.71 |
| 3 | 774 | 67 | 740.5 | 69 | 0.09 | 0.91 | 0.65 |
| 4 | 638 | 85 | 595.5 | 29 | 0.05 | 0.95 | 0.62 |
| 5 | 524 | 82 | 483.0 | 44 | 0.09 | 0.91 | 0.56 |
| 6 | 398 | 50 | 373.0 | 50 | 0.13 | 0.87 | 0.49 |

Although the data sets gathered and analyzed cover the time period of a merger, a significant impact of the merger on the probability of death of an application system was not observable within the analyzed time period. An interview with the industry partner regarding this expected impact revealed that the impact seems to be delayed about three

years after the actual merger took place. This hypothesis is backed by the planned architecture of company B, which predicts a large number of changes including replacements of applications for the year 2009.

4.3 Company C

Company C is a major financial service provider in Switzerland primarily focusing on standardized retail banking and transaction processing. All EA levels from business architecture to IT infrastructure in [WF06] are managed by broad, defined architecture management processes. An initiative was started to manage business and organizational architecture artifacts by the IS department architecture team as well. However, this has been dropped in favor of managing all business related artifacts by an explicit business architecture management team itself, which forms an organizational unit attached directly to the CEO in the business development department. The alignment of business and IS architectures is explicit and facilitated by a personal interweavement by having former IS architects included in the business architecture unit. Core EA artifacts are captured in an EA repository including timestamps for tracking architectural change. However, only the most requested artifacts are regularly modeled due to the high cost of keeping models up to date.

The data set of company C was extracted from the EA repository via Excel files. It includes applications as well as their clustering in domains. These artifacts are the core elements in the company's EA. For the life table analysis we only considered applications, such as a Data Warehouse, a Card Transaction System, or an SAP system.

The extracted snapshots from the EA repository have multiple timestamps from different dates in the years 2006 through the beginning of 2009. We analyzed six time periods each representing six months of observed time in the years 2006, 2007, and 2008 (cf. Table 4).¹

¹ In order to align the life table with the findings from companies A and B the time periods are denoted on the same scale, resulting in decimal numbers for company C.

Table 4: Life table for applications of company C

| Time period (Year) (x) | Number at start (n_x) | Withdrawn during year (w_x) | At risk (r_x) | Deaths (d_x) | Prob. of death (q_x) | Prob. of surviving period x (p_x) | Cum. prob. of surviving the period x (P_x) |
|----------------------------|---------------------------|---------------------------------|-------------------|------------------|--------------------------|---|--|
| 1.0 | 189 | 0 | 189.0 | 28 | 0.15 | 0.85 | 0.85 |
| 1.5 | 161 | 4 | 159.0 | 0 | 0.00 | 1.00 | 0.85 |
| 2.0 | 161 | 6 | 158.0 | 29 | 0.18 | 0.82 | 0.70 |
| 2.5 | 132 | 1 | 131.5 | 6 | 0.05 | 0.95 | 0.66 |
| 3.0 | 126 | 5 | 123.5 | 2 | 0.02 | 0.98 | 0.65 |
| 3.5 | 124 | 12 | 118.0 | 7 | 0.06 | 0.94 | 0.61 |
| 4.0 | 117 | 10 | 112.0 | 8 | 0.07 | 0.93 | 0.57 |

The overall number of applications captured in the EA remains quite constant over the time period analyzed. However, there are some high losses (d_x) in the periods 1 and 2. This indicates a volatile application landscape at company C concerning the artifacts contained in the EA repository. After the time periods that we analyzed, the cumulative probability of an application to survive the 6 periods was at 0.57, i.e. in average 57% of the applications survive 4 years.

5 Results and discussion

The above case studies demonstrate the feasibility of applying the life table method to assess life spans of EA artifacts via calculating the probability of particular applications to survive a certain number of years. In doing so, data was gathered via EA models within the different companies, presuming that they precisely reflect the survival behavior of the real applications. While observing the life span of applications within an enterprise, one is confronted with similar conditions like in medical studies: subjects enter and leave the study at different times and thus give insights into their life spans. As we were only concerned with externally observable behavior regarding system emergence and decay, we treated the applications as *black boxes*, i.e. we did not analyze their internal changes. Moreover, the findings from the three case studies give quantitative evidence about approximate life spans of the applications listed in EA repositories. Figure 1 summarizes the results from the three case studies by depicting the three *survival curves* that are generated by the calculated cumulative probability of surviving x years. The graphs are drawn stepwise based on the calculations from the life tables. On the one hand, this reflects the gaps in the data sets concerning the time periods analyzed, with abrupt changes in the probabilities. On the other hand, the comparison of the three data sets shows that the results are quite similar for applications in all three companies: The probability to survive for 4 years seems to be around 60% on average.

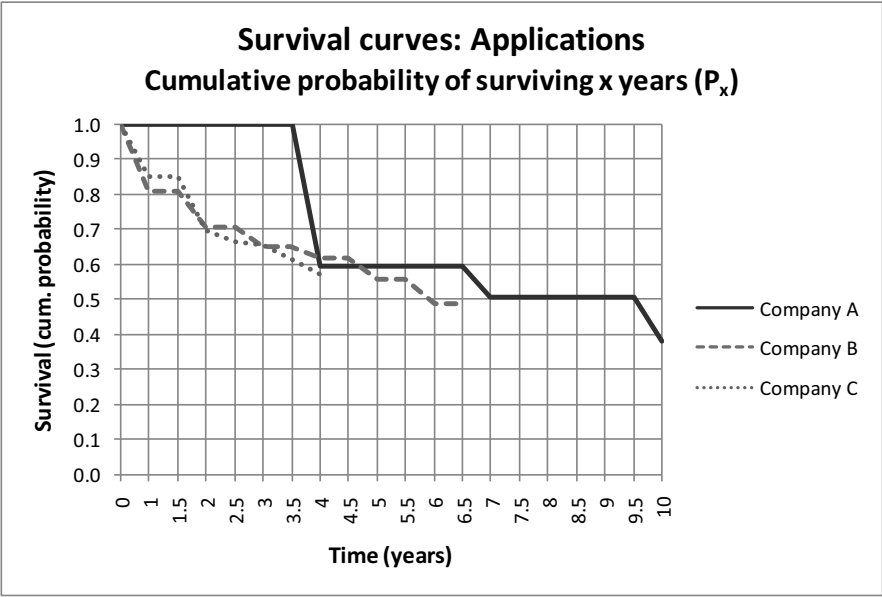


Fig. 1: Survival curves for the case studies

The results provide indications on how to design an EA model maintenance process that is adjusted to the decay rates of applications. A cumulative probability of 60% implies that more than half of the information on applications covered in an EA model becomes obsolete after four years without updates. Consequently, an effective EA model maintenance process will include a sampling rate smaller than four years in order to ensure timeliness of the information and efficiency of the related resource management at the same time.

Drawing more elaborate conclusions from these results is difficult because of several limiting factors. First of all, the limited data sets available hinder a thorough analysis. In the three case studies reliable data was only available for applications. This led to rather exemplary results concerning the survival of EA model artifacts. Also, a detailed analysis is only possible, if the observations underlying the life tables are conducted with equal intervals. While this is often the case in the medical setting from where the life table method has been adopted, it is not necessarily the case in the realm of EA. Indeed, in our investigation, only companies B and C proved mature enough as to have regular and reliable data on a yearly (or even better) basis. Company A could only provide more coarse datasets. Due to this fact, the results of the study necessarily remain on a rather aggregate level.

While our findings result in quantitative figures it is hard to assess these figures without a proper frame of reference, or a baseline. Such a baseline would need to take into account several phenomena, including:

1. Some extensive shifts in the number of applications can be traced back to external events such as mergers and acquisitions (as described above, this will be the case for company B in the near future) or organizational restructurings.
2. The notion of *industry clockspeed*. It has been suggested that different industries have different rhythms in their use of IT [Fi96], and such a phenomenon might be reflected in the data analyzed. At the single point of comparison (4 year period) the analysis results of the different case studies refute the industry clockspeed hypothesis. Nevertheless, the number of companies studied is much too small to draw any final conclusions.

There are many possible factors that add *noise* to the data sets in the present study, or make them hard to interpret. The following discussion aims to address those issues:

The influence of external or contingency factors has not been considered and is thus not reflected in the data. For example, the results are currently not related to the use of specific EA management approaches or methods in the respective companies. Such considerations might increase the commensurability of the data. Other external factors, like mergers and acquisitions, new legislation, or Y2K-phenomena might also influence the findings in detail.

A related uncertainty concerns the previously mentioned question of a baseline. *What is the normal state?* A model aiming to support decisions on optimal sampling rates needs to establish the context in which it is applicable. Another way to phrase this is to ask what is being described. *Is it only changes due to old technology, or also changes due to how organizations work, including the recurring application consolidations?* This means that factors such as the following should be either deliberately included or deliberately excluded:

- Exogenous (as seen from the IT department scope) factors, e.g. mergers and acquisitions, changes in legal regulations
- Endogenous (as seen from the IT department scope) factors, e.g. application consolidation
- Changes in scope of data over time (e.g. models from different time periods might have used different definitions)
- Poor data quality affects model credibility (e.g. very high numbers of withdrawals)
- Reliability of data, i.e. how close to reality the constituent data sets are. For instance, sometimes changes in taxonomies make data sets seem very different, even though the actual landscape they are describing is more or less the same.

In the present studies, no effort has been made to deliberately neither include nor exclude these factors. The construction of reliable statistical procedures for such data cleaning requires, however, more data than is presently available.

6 Conclusions and future work

This paper described a method for analyzing the rate of change of EA artifacts. Specifically, the method has been used to analyze the rate of change of the application portfolios at three different companies. The results were presented in the form of survival curves showing the likelihood of survival of applications in the respective companies. Thereby, we presented a contribution to the field of EA model management and maintenance. The long-term aim is to be able to find optimal sampling rates of model updating, based on the underlying rate of change of the EA artifacts, which the EA models should describe. A final version of such a method would include a trade-off between the benefit of a high sampling rate and the cost of frequent data collection.

In more or less all the cases, longer time series would have been conducive to properly account for contingency factors and find the *normal* state of the rate of change of applications. One indicator of an appropriate length of the time series is the cumulative probability of survival, P_x . When $P_x = 0$ there is obviously no use for a longer time series. While reaching zero is unlikely in most organizations, a low value of P_x could, however, be an indicator that enough data is collected. In this study, the lowest cumulative probability of survival was 37%, which would indicate the need for more data and longer time periods of observation for the case studies.

The presented study focused on investigating the rate of change of applications. To be useful, analyses of change rates need to cover other EA artifacts as well, e.g. business processes or interfaces. A more elaborate analysis method could then be utilized to determine different areas within the EA, which change at different rates and hence require appropriate sampling rates. Such areas might be formed by the types of artifacts, e.g. applications and business processes or different *subtypes* within one artifact type, e.g. different types of applications. Equally such areas could well be defined in other manners, such as for instance all entities connected to a critical business process. Furthermore, an analysis method could be used to investigate tangible EA artifacts, like applications, and their relationships to their costs in order to support investment decisions.

To generalize the analyses of rates of change to other companies and other lines of business there is a need not only for more data from other companies to populate the life tables for different EA artifacts, but probably also a richer model of how contingency factors may affect the change rates of the EA artifacts mentioned above. This would probably entail defining probability distributions in which parameters can be set. Analysis of survivability curves could be a first step towards defining such distributions. Based on the result of more detailed analyses, the process of EA model maintenance could be refined with information on optimal sampling rates for distinct EA artifacts.

References

- [AW09] Aier, S.; Winter, R.: Virtual Decoupling for IT/Business Alignment – Conceptual Foundations, Architecture Design and Implementation Example. In: Business & Information Systems Engineering 51 (2009) 2, pp. 150–163.

- [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.
- [BK05] Bittler, R. S.; Kreizmann, G.: Gartner Enterprise Architecture Process: Evolution 2005. Gartner Inc., Stamford, CT 2005.
- [Bl00] Bland, M.: An Introduction to Medical Statistics. 3 edition, Oxford University Press 2000.
- [BPH65] Barlow, R. E.; Proschan, F.; Hunter, L. C.: Mathematical Theory of Reliability. John Wiley & Sons, New York 1965.
- [Ca03] Carroll, K. J.: On the use and utility of the Weibull model in the analysis of survival data. In: *Controlled Clinical Trials* 24 (2003), pp. 682–701.
- [Ci01] Chief Information Officer Council: A Practical Guide to Federal Enterprise Architecture, Version 1.0. 2001.
- [Dv01] Department of Veterans Affairs: Enterprise Architecture: Strategy, Governance, & Implementation. 2001.
- [FAW07] Fischer, R.; Aier, S.; Winter, R.: A Federated Approach to Enterprise Architecture Model Maintenance. In: *Enterprise Modelling and Information Systems Architectures* 2 (2007) 2, pp. 14–22.
- [FCH83] Freeman, J.; Carroll, G. R.; Hannan, M. T.: The Liability of Newness: Age Dependence in Organizational Death Rates. In: *American Sociology Review* 48 (1983) 5, pp. 692–710.
- [Fi96] Fine, C. H.: Industry Clockspeed and Competency Chain Design: An Introductory Essay. In: *Proceedings, 1996 Manufacturing and Service Operations Management Conference* 1996.
- [GG01] Gavrilov, L. A.; Gavrilova, N. S.: The Reliability Theory of Aging and Longevity. In: *Journal of theoretical Biology* 213 (2001), pp. 527–545.
- [Go25] Gompertz, B.: On the Nature of the Function Expressive of the Law of Human Mortality, and on a New Mode of Determining the Value of Life Contingencies. In: *Philosophical Transactions of the Royal Society of London* 115 (1825), pp. 513–585.
- [If99] IFIP–IFAC: GERAM: Generalised Enterprise Reference Architecture and Methodology, Version 1.6.3. IFIP–IFAC Task Force 1999.
- [JE07] Johnson, P.; Ekstedt, M.: Enterprise Architecture: Models and Analyses for Information Systems Decision Making. Studentlitteratur 2007.
- [La05] Lankhorst, M.: Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer, Berlin, Germany 2005.
- [Mo09] Moser, C. et al.: Some Process Patterns for Enterprise Architecture Management. In: *Proceedings, Workshop on Patterns in Enterprise Architecture Management (PEAM2009)*, Bonn 2009.
- [Og09] The Open Group: TOGAF (The Open Group Architecture Framework) Version 9. 2009.
- [RWR06] Ross, J. W.; Weill, P.; Robertson, D. C.: Enterprise Architecture as Strategy. Harvard Business School Press, Boston, Massachusetts, USA 2006.
- [SH93] Spewak, S. H.; Hill, S. C.: Enterprise Architecture Planning – Developing a Blueprint for Data, Applications and Technology. John Wiley & Sons, New York 1993.
- [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, pp. 1–8.

Evaluating Enterprise Architecture Management Initiatives – How to Measure and Control the Degree of Standardization of an IT Landscape

| | | | |
|------------------------------|--------------------------------|-----------------------------|----------------------------|
| Matthias Brückmann | Klaus-Manfred Schöne | Stefan Junginger | Diana Boudinova |
| ZIVIT | ZIVIT | BOC Germany | BOC Germany |
| Wilhelm-Fay-Str. 11 | Wilhelm-Fay-Str. 11 | Voßstr. 22 | Voßstr. 22 |
| D-65936 Frankfurt | D-65936 Frankfurt | D-10117 Berlin | D-10117 Berlin |
| Germany | Germany | Germany | Germany |
| matthias.brueckmann@zivit.de | klaus-manfred.schoene@zivit.de | stefan.junginger@boc-de.com | diana.boudinova@boc-de.com |

Abstract: An important objective of nearly all EAM initiatives is the standardization and consolidation of the IT landscape. This can be seen on several levels, e.g. on the application landscape level, the software level, the hardware level, etc. This paper presents a stepwise approach for consolidating IT landscapes on the technology, software and hardware level which considers the specifics of large IT organizations and is implemented in one of the largest IT service providers of Germany's public civil administration. It is based on metrics which use concepts from fuzzy logic. In particular, these metrics allow measuring the degree of standardization of parts of as well as of the overall IT landscape.

1 Introduction

The steadily growing complexity of IT landscapes, the increasing speed of changes of business models, business process and organizational structures in many business sectors, and the high pressure for cost reductions in the IT area have lead to the intensification of Enterprise Architecture Management (EAM) activities in the past few years. There are numerous frameworks for EAM [Sc06], [FE09], [TO09]. However, there is relatively little literature how to evaluate the success of EAM initiatives. For this, metrics derived from the EAM objectives are needed.

A common issue in EAM is the enormous heterogeneity of the IT landscapes in many organizations. There are used a variety of IT objects (ITO): hardware platforms, database products, operating systems, application servers, development tools, programming languages, etc. A reduced set of such ITO leads to lower operation, administration and maintenance costs, improved security, reliability and development time, and more cost-effective delivery of services [IT07], [RW06], [TO09]. Therefore, standardization, or in other words the shift to a consolidated IT landscape, is an important EAM objective.

Standardization however comes at the price of fewer choices of IT solutions [RW06]. Furthermore, Gartner warns against "over-standardization", i.e. focusing too much on defining standards and neglecting business value, and the forceful top-down manner of implementing it [Ga09]. Therefore, the consolidation of ITO needs to be done in a stepwise manner and in consideration of business requirements.

In this paper we present an approach for controlling and measuring the degree of standardization of an IT landscape regarding ITO which considers the specifics of large IT organizations. The presented metric uses fuzzy logic concepts and is based on a simple conceptual model for representing IT landscapes and a stepwise standardization process for the IT landscape consolidation. The presented approach has been implemented at the Center for Information Processing and Information Technology (Zentrum für Informationsverarbeitung und Informationstechnik – ZIVIT), one of the largest IT service providers in Germany's public civil administration [ZI09]. ZIVIT is designing, hosting and maintaining more than 300 applications and 50 new application projects used in the public administration, which involves the management of an extensive IT landscape of over 2,000 ITO. The authors of this paper are practitioners – two are enterprise architects at ZIVIT and two are with BOC, a company providing EAM tools and solutions (ADOit, ADOben) and EAM consulting [BO09].

The remainder of this paper is organized as follows: In chapter 2 metrics for measuring Enterprise Architectures (EA) are discussed. Then, in chapter 3 a stepwise process for standardizing ITO is described which is based on a conceptual model for IT landscapes and a lifecycle model of ITO. Chapter 4 presents metrics for measuring the degree of standardization of an IT landscape. Chapter 5 summarizes practical experiences of applying the presented metrics. Chapter 6 concludes with an outlook for future work and research topics.

2 Related Work

In EAM there are no universally accepted metrics. Of course, in most organizations there are defined at least some EA metrics. However, there are only few publications which describe such metrics and experiences in applying EA metrics [AD05], [RG07], [Va08]. Methods how to define metrics for EA are described in [IT07], [Fr08] and [He08]. It has to be noted that there is also a research community on "very large business applications" [Gr07] which has a significant overlapping with EAM. It focuses on methods for the evolution of existing application landscapes and legacy systems. For example, in [Mu08] metrics for the so-called "Managed Evolution" of large business applications are described which are used to decide on the further development of legacy systems.

The issue of standardization of IT landscapes is not widely treated in the literature. Its cost-cutting benefits are briefly discussed in TOGAF [TO09] and in the IT Infrastructure Library (ITIL) [IT07]. In [RW06] "standardized technology" is considered the second stage of architecture maturity readily embraced by companies due to its many benefits as presented in the introduction of this paper. However, the authors are not aware of any publications on how to measure and control the degree of standardization of an IT landscape. ITIL defines technology metrics, which however refer to the performance and availability of components and not to standardization [IT07]. The concept of standardization is based on the reuse of ITO in different applications and is thus related to the area of reuse management. [MC07] provides a literature review of reuse metrics and a report of their application in industrial studies, which are however focused on reuse in software development. Of course, most organizations have "technology lists" which define which software and hardware products shall be used. There often exist nevertheless neither defined processes how these technology lists are updated in a controlled manner, nor metrics which measure compliance. With the approach presented in the following chapters we aim to overcome these deficits.

3 Process for Standardizing ITO

One of the main competencies of ZIVIT is the design and maintenance of new applications used in the public administration. In this respect, assuring that these applications use only standardized ITO, i.e. ITO which are considered "standard" within the organization, is an important objective of EAM at ZIVIT. Before the standardization process and the standardization degree metrics can be discussed, it is necessary to present the used conceptual model for IT landscapes and a lifecycle concept for ITO.

3.1 Conceptual Model

Figure 1 depicts the conceptual model used at ZIVIT upon which the standardization metric is based. Applications use ITO – we distinguish between software, hardware and technology types of ITO. In the literature, software and hardware are often considered sub-elements of technology. However, the concept of "technology" how it is used here allows representing software architecture patterns. Additionally, there are guidelines in Germany's public administration (see below) which can be described best by introducing an additional concept to software and hardware.

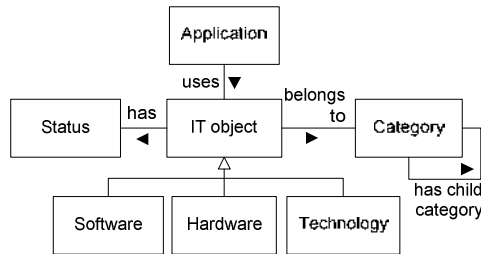


Figure 1: Conceptual model of IT landscapes

By software we mean system software such as operating systems, database products, compilers, en-/decryption software, application servers, etc. Hardware refers to physical components such as servers, workstations and storage devices. By technology we understand software architecture patterns (2/3 tier application, web application, etc.) and standards such as application protocols, digital preservation formats (e.g. XML), graphic formats, technology for information processing (e.g. HTML, XSL, JSP), programming languages (e.g. COBOL, PL/SQL), etc. Germany's public administration has for example defined a list of architecture patterns and technologies which shall be used in e-Government applications (SAGA, Standards and Architectures for e-Government Applications) [BM08].

Each type of ITO contains an unsorted set of elements. This set is broken down into function-oriented categories (subsets) which can be further refined into child categories. Each refinement step creates a new category level in the tree. Through this decomposition ITO can be represented as the leaves – at the lowest level – of a tree of categories. Figure 2 shows an example for the software category DBMS. Examples for ITO are products such as Oracle 9i, Oracle 10g, Oracle 11g, SQL Server 2005 etc.

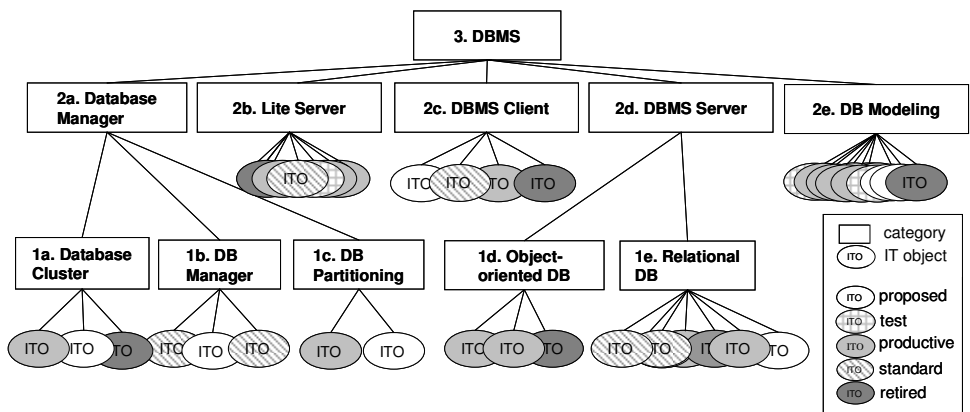


Figure 2: Category tree – Example for software

Each ITO has a status which describes in which phase of its lifecycle it is located. There are five statuses defined at ZIVIT explained in the following subsection.

3.2 Lifecycle of an ITO

In order to understand the concept of standard ITO upon which the calculation of the standardization metrics is based, it is important to describe the lifecycle of ITO at ZIVIT. The possible transitions between the five defined statuses are visualized in figure 3 and subsequently explained in the description of the standardization process.

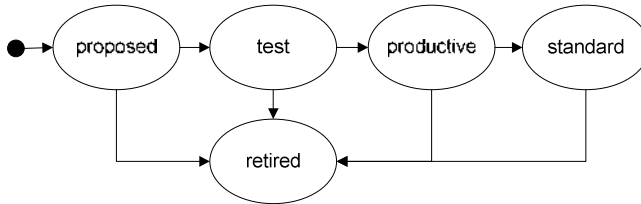


Figure 3: Transition between ITO statuses

Proposed: The status "proposed" suggests that it has been assessed that an existing ITO within a category does not meet completely the requirements or that a new ITO is available which promises advantages in comparison to existing ones. Alternatively, there might be seen the need for a new category with new ITO.

Test: After a proposal has been approved, the ITO enters the test phase. Depending on the test results the ITO with status "test" can either be admitted to production (status "productive") or rejected (status "retired").

Productive: Once an ITO enters into use in an application, it obtains the status "productive". Productive ITO are such that have successfully completed the test phase and are considered necessary or appropriate for one or more applications. A productive ITO can either be promoted to status "standard" or can be retired if it does not fit the long-term IT strategy of the organization or could not prove itself in production. The status "productive" has been introduced due to the size of ZIVIT and its IT landscape. In other organizations it might be common to transition directly from "test" to "standard" respectively to "retired".

Standard: If the benefits of an ITO and its alignment with the IT strategy could be proven in production, the ITO is promoted to standard. ITO with status "standard" are officially released and made available for all applications. Standard ITO are approved as such by the management and are obligatory in applications with the same requirements.

Retired: This status can be obtained after all possible statuses but only at the discretion of the management body (except in the case when a proposed ITO is rejected). Retired ITO are not to be used in new applications. In existing applications they can continue to be used but need to be reviewed at the time of the next release. ITO with status "proposed" or "test" which are rejected receive directly the status "retired". In such cases is necessary to maintain documentation of such rejected proposals in order to prevent redundant evaluations of ITO in the future. Rarely it happens that a new version of rejected ITO has to be reevaluated, e.g. due to market changes. Then, there is created a new ITO (with status "proposed").

3.3 Standardization Process

In order to ensure the involvement of different organizational roles in the standardization process and thus secure their acceptance, a standardization process has been established at ZIVIT which controls the transitions between the life-cycle statuses of ITO and is depicted in figure 4. From our experience well-governed IT organizations have similar processes in place.

The standardization process at ZIVIT is activated when a trigger such as new IT trends, customer requirements, changes to management strategy, etc. leads to a request for a new ITO in a written form submitted along with a statement of motivation to the enterprise architects. Such a change request can be submitted by any organizational role. After the proposal has been acknowledged by the enterprise architects, the ITO receives the status "proposed".

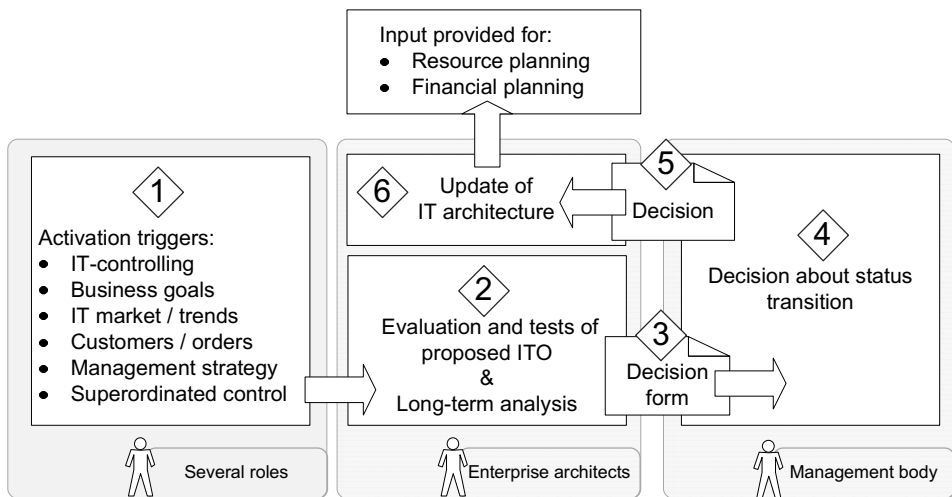


Figure 4: Standardization process

The main part of the process is the analysis phase in which the expected long-term effects of decisions are evaluated and tested. The following questions are considered: Is there an actual need for a new ITO or can the demands be met by an already existing object or is the request of a strategic importance, e.g. is the new ITO expected to have far-reaching consequences for the entire EA? The proposed object is reviewed by the enterprise architects and can be either marked as relevant which changes its status to "test" or is rejected leading to the status "retired". The proposed ITO, along with alternatives when required, are tested and evaluated with the help of a test results template.

After tests are completed, a decision form is created and submitted to the management body by the enterprise architects. Each status transition after the test phase is to be determined in the decision-making process. In case of positive test results, the selected ITO applies for a status "productive". If tests fail, the ITO is recommended for rejection - status "retired".

If the ITO has proven itself – for this, there is a detailed list of criteria –, it is promoted to status "standard", again with a decision form. If an ITO is no longer supported or maintained by its vendor or due to some reason such as obsolescence is no longer to be further used in new applications, a decision form is used to request the new status "retired" for this ITO.

Each time that a decision has been made by the management body based on the information provided as input by the analysis phase and communicated back to the enterprise architects, the IT architecture is updated and the update is documented in the repository. It is important that changes to the IT architecture further provide input for and are evaluated in terms of their effects on the resource and financial planning.

4 Metrics for the Standardization Degree

As specified in the conceptual model explained in section 3.1, ITO are grouped in categories which are further broken down in child categories (see figure 2). In order to propagate the standard status of ITO to the category level a number of conditions need to be introduced:

Postulate 1: There is no mixing of ITO and categories as children of a single category.

Postulate 2: A hierarchy of categories is built upon the level of the ITO. This hierarchy is described with the category tree as presented in section 3.1

These two postulates imply:

Corollary 1: ITO are located always at the lowest level of the category hierarchy. The ITO are the leaves of the category tree.

The hierarchical categorization of ITO allows determining the standardization degree metric for each category in the tree based on its elements. If this metric is applied to the root category, one would obtain the standardization degree for an entire type (i.e. software, hardware, technology). Calculating the standardization degree requires applying the concept of standardization to the category tree. The further standardization of elements of the category tree is defined in the following section 4.1.

4.1 Definition of Standardization

Definition 1: A category *K* containing ITO is called standardized if and only if the number of child ITO with status "standard" is one or two, i.e.:

$$1 \leq \#\{ITO \in K \mid st(ITO) = \text{"standard"}\} \leq 2$$

with $st()$ being the function which returns the status of an ITO (or category).

Allowing two standard ITO within a category has its justification. Very often a single product cannot fulfill the particular requirements of all applications which use an ITO of this category. Furthermore, variability in the costs of the ITO (license costs, maintenance costs) could lead to differing economic efficiency in different applications. For example, in the category of signature methods products affirmed by the law regarding electronic signatures in Germany (SiG) [BJ09] are needed for qualified digital signatures, whereas for technical signatures JAVA components are sufficient. Nevertheless, the decision to allow two standard ITO within a category is ZIVIT-specific and might be different in other IT organizations.

Definition 2: A category K^n on level n containing categories K^{n-1} of level $n-1$ is called standardized if and only if its child categories of level $n-1$ are standardized, i.e.:

$$\#\{K^{n-1} \in K^n\} = \#\{K^{n-1} \in K^n \mid st(K^{n-1}) = \text{"standard"}\}$$

The notation K^n signifies the level at which a category K is located, i.e. the categories from level $n-1$ are elements of the categories of level n .

4.2 Standardization Degree of Categories

The standardization degree is a value between 0 and 1, similar to the fuzzy logic concept according to which variables can have an intermediate membership value [Za65]. The rationale behind the definitions introduced below is the following: Even if a category is standardized, ITO with status "productive" can weaken this standardization if these ITO are used in "too many" applications.

Definition 3: The standardization degree $SD(K)$ of a category K containing ITO is defined as follows:

$$SD(K) := \begin{cases} \frac{\sum_{ITO \in K} g_{ITO} \delta_{ITO}}{ST_{Sub} + Prod_{Sub}}, & \text{if } 1 \leq ST_{Sub} \leq 2 \\ 0, & \text{otherwise} \end{cases}$$

where $ST_{SUB} := \#\{ITO \in K \mid st(ITO) = \text{"standard"}\}$ and

$$Prod_{SUB} := \#\{ITO \in K \mid st(ITO) = \text{"productive"} \text{ and } g_{ITO} > 0\}$$

The parameter δ_{ITO} is defined for each ITO of the category K such that:

$$\delta_{ITO} := \begin{cases} 1, & \text{if } st(ITO) = \text{"standard"} \text{ or } st(ITO) = \text{"productive"} \\ 0, & \text{otherwise} \end{cases}$$

The factor δ_{ITO} retrieves simply the status of an ITO and is equal to 0 for all objects which are with status "proposed", "test" or "retired". The reasoning behind the decision for not including them in the metric calculation is that ITO with these statuses have no direct effect on standardization or the transition between statuses "productive" and "standard". Hence, "standard" and "productive" are the only statuses which are of importance for the usage of an ITO in operations.

The parameter g_{ITO} is defined for each ITO as follows:

$$g_{ITO} := \begin{cases} 1, & \text{if } st(ITO) = \text{"standard"} \\ g_{p_{ITO}}, & \text{if } st(ITO) = \text{"productive"} \end{cases}$$

The parameter $g_{p_{ITO}}$ is defined for each productive ITO as follows:

$$g_{p_{ITO}} := \begin{cases} 0, & \text{if } \frac{\#\{applications \mid ITO \in application\}}{\#\{applications\}} \leq TV \\ \frac{\#\{applications \mid ITO \in application\}}{\#\{applications\}}, & \text{otherwise} \end{cases}$$

TV is a threshold value above which an ITO with status "productive" enters into the calculation as it means that the ITO is in use in a relatively high number of applications of the overall IT landscape and can be considered quasi-standard. The value of TV is set based on the business requirements, advisably by the enterprise architects. We use the value 0.05.

The factor g_{ITO} determines whether the contribution of an ITO is full – when its status is "standard" – or fractional when "productive". Productive ITO which are not used in more than a certain percentage of the total number of applications within the overall IT landscape are not considered, i.e. g_{ITO} is 0. For productive ITO which are considered as quasi-standard due to their extensive usage g_{ITO} is the percentage of applications of the overall IT landscape in which these productive ITO are used.

It is obvious that the standardization degree $SD(K)$ of a category K containing one or two ITO with status "standard" and no ITO with status "productive" is 1. The same applies if there are contained ITO with status "productive" which are only relatively rarely used in applications. The presence of quasi-standard ITO reduces the standardization degree of a category to indicate that the category contains productive ITO which are not with status "standard" but are still used by a significant number of applications.

Definition 4: The standardization degree $SD(K^n)$ of a category K^n containing categories K^{n-1} of level n-1 is defined recursively as follows:

$$SD(K^n) := \frac{\sum_{K^{n-1} \in K^n} SD(K^{n-1})}{\#\{K^{n-1} \mid K^{n-1} \in K^n\}}$$

So, $SD(K^n)$ of a category K^n of level n containing categories K^{n-1} of level n-1 is the average of the standardization degrees of its containing categories.

It can be easily shown that $0 \leq SD(K) \leq 1$ for all categories K .

4.3 Further Metrics

For a complete control of the standardization process, additional metrics are needed, which are defined below.

The standardization degree does not capture how widely a standard ITO are used. Therefore we introduce the spread of an application/application landscape.

Spread of an application/application landscape: The ratio of the standard ITO of an application to the total number of ITO used in the application is called the spread of an application. A statement about the entire application landscape can be made by calculation the average spreads of the applications:

$$\frac{\sum_{i=0}^n \frac{\#\{ITO_s \in application_i\}}{\#\{ITO \in application_i\}}}{n}$$

where ITO_s :=ITO with status "standard" and n :=number of applications.

To measure the potential reuse rate of an ITO, we introduce the following metric:

Possible spread of an ITO: The ratio of the number of applications in which an ITO is used to the total number of the applications in which the ITO could be potentially used is called possible spread of an ITO, i.e.:

$$\frac{\text{Number of applications in which the ITO is in use}}{\text{Total number of applications in which the ITO could be used}}$$

This shows the actual usage of an ITO in relation to its possible one. Hence this metric is similar to the reuse rate presented in [MC07]. In the same way the spread of all ITO of a category can be calculated.

In particular, when introducing the standardization process, there will be probably a high number of ITO with status "productive". Therefore, it is of interest to measure how many of them are quasi-standard:

Number of quasi-standard ITO: This metric represents the number of ITO in the landscape that have a value for $g_{\pi o}$ above the defined threshold value, as it was introduced in definition 3 in section 4.2.

The number of quasi-standard ITO complements the standardization degree in measuring the success of the standardization process. Additionally, for a single quasi-standard ITO, the standardization process might be triggered (see section 5.1).

5 Example and Practical Experiences

This chapter presents first an example of the application of the standardization degree metric. Then, experiences and benefits of the application at ZIVIT are described.

5.1 Exemplary Application of the Standardization Metric

Figure 5 shows the standardization degrees for the category tree from figure 2. This means that the category "DBMS" is 47.4% standardized. This implies that the categories on the lower levels need to be investigated in order to account for the low degree of standardization. Four non-standardized categories are found – three at the lowest category level – 1a, 1c and 1d, and a fourth one at the second level – 2e. A standardization degree of 0 means that in these categories either no or more than two standard ITO are in use. An issue that comes to being here is the priority of resolution of such multiple cases. For this prioritization weights of categories as described in section 5.2 can be used.

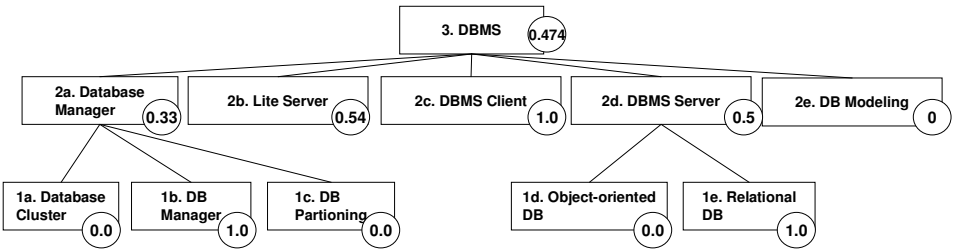


Figure 5: Application of standardization metric to a category tree

Furthermore, category 2b has the value of 0.54. The value between 0 and 1 suggests that there is at least one quasi-standard ITO. Assume, there is an ITO with status "productive" which is used in more than 5% of the total number of applications. The task of the enterprise architect in this case is to review the quasi-standard ITO and its usage in applications in order to investigate the reasons behind the fact that given that a standard ITO already exists in a category, such a high number of applications are using another non-standard ITO from this category.

5.2 Experiences and Benefits of Applying the Standardization Metric

The presented approach has been fully implemented at ZIVIT. Of course, the implementation and operation of the standardization process required significant effort. At ZIVIT it needed two enterprise architects for one year to model the 2,000 ITO with all their attributes. ITO at ZIVIT and their hierarchical categorization are catalogued and kept up-to-date in a repository with the EAM tool ADOit [AD09], [Ju08]. For example, more than 1,500 software objects in 10 main and 150 child categories are defined. The top software categories are defined according to the different steps of the software development process, i.e. requirements engineering, functional design, technical design, build, test, deployment. This leads to categories such as requirements engineering and specification tools, development tools, run-time libraries/tools, databases, execution environments, etc.

The relation between applications and ITO is maintained by the application owners (approximately 100 at ZIVIT). To ensure that the application owners really do this maintenance well-defined processes and some organizational change management is needed. However, this effort is more than compensated by the enormous cost reductions in terms of human resources, administration effort in production, and last but not least, license and maintenance costs. Imagine for example the effect of a reduction of fictive 10 million Euros yearly maintenance costs by 5% (which is not an unrealistic reduction rate). The final aim is to achieve a standardization degree of 1 for the entire IT landscape. Of course, this is a vision which will never be reached in large IT organizations. However, on the way towards this ideal state, the historical record of the development of the value of the metric is an indication for the results of the process. Summarizing, the presented approach proved its value at ZIVIT. Actually, it was already extended by representing expiration dates of the maintenance levels of software and hardware ITO to enable proactive measures for replacing ITO.

Something we have not done yet due to the high effort required but we are seriously considering is the definition of weights for categories of ITO. A category of strategic value, e.g. database products, has a different importance than the category of text editors because the latter is most likely much easier to replace. This idea would require adapting definition 4 of section 4.2 by assigning a weight between 0 and 1 to each category, according to the effort for replacing a contained ITO by another one from this category. In this way, the standardization degree of a category would still have a value between 0 and 1. However, the standardization degrees of the contained categories will be weighed accordingly.

As mentioned in the introduction, over-standardization is a practice in EAM that should be avoided. Over-standardization should not be understood as an excessive degree of standardization of the IT landscape but as a tendency to force standards on the organization without consideration of the business needs. The introduced standardization metric contributes towards the avoidance of over-standardization through exposing quasi-standard ITO in its calculation. As defined in section 4.2, quasi-standard ITO are such that are used in a high number of applications without having the status "standard". Their presence indicates a possible discrepancy between business needs and established standards. Their revision and possible promotion to standard objects is an essential step towards aligning business needs with the IT landscape.

6 Outlook

The presented approach has been developed for large IT organizations using several hundred or even several thousand ITO. However, by adapting the parameters of the metrics the presented approach works for smaller IT organizations too. It could be defined for example that only one standard ITO per category is allowed instead of two (compare definition 1 in section 4.1). It is crucial to be aware that the application of the presented metrics relies upon the availability of always up-to-date EA data. From our experience this is one of the major obstacles for the success of EAM. Here we see a large research potential regarding EAM processes. Ideas and concepts how to design and implement EAM processes are presented for example in [Ke07] and [Mo09].

From a practitioner's point of view we see the need for more extensive research on EAM metrics. An interesting approach would be for example to apply data warehouse concepts on EA data. Additionally, we see a large potential in applying the "EAM patterns approach" [Bu08] on EAM metrics. This would allow to share experiences about EAM metrics in a convenient way by stating in which situations which metrics can (or should) be applied.

References

- [AD05] Aier, S.; Dogan, T.: Indikatoren zur Bewertung der Nachhaltigkeit von Unternehmensarchitekturen. In: Ferstl, O. et al. (Eds.): Wirtschaftsinformatik 2005: eEconomy, eGovernment, eSociety, Bamberg, Physica, Heidelberg, 2005, pp. 607-626.
- [AD09] EAM tool ADOit, www.boc-group.com/adoit (access 2009-04-25).
- [BJ09] Bundesministerium der Justiz: Gesetz über Rahmenbedingungen für elektronische Signaturen, http://bundesrecht.juris.de/sigg_2001/index.html, (access 2009-07-29).
- [BM08] Federal Ministry of the Interior (Germany): Standards und Architekturen für E-Government (SAGA), www.kbst.bund.de/saga (access 2009-05-21).
- [BO09] BOC Homepage, www.boc-group.com, (access 2009-04-25).
- [Bu08] Buckl, S. et. al.: Enterprise Architecture Management Pattern Catalog. Release 1.0, Garching b. München, Germany 2008, <http://srvmatthes8.informatik.tu-muenchen.de:8083/file/EAMPatternCatalogV1.0.pdf> (access: 2009-05-21).
- [FE09] Federal Enterprise Architecture, <http://www.whitehouse.gov/omb/e-gov/fea/> (access 2009-05-21).

- [Fr08] Frank, U. et al.: Designing and Utilising Business Indicator Systems within Enterprise Models – Outline of a Method. In: Loos, P. et al. (Eds.): Modellierung betrieblicher Informationssysteme (MobIS 2008), GI, Bonn, Vol. 141, Lecture Notes in Informatics, 2008, pp. 89-105.
- [Ga09] Gartner Research (Ed.): Thirteen Worst Enterprise Architecture Practices. Report No. G00164424, 2009-01-28.
- [Gr07] Grabski, B. et al.: Very large business applications. In: Informatik Spektrum. Vol. 30, No. 4, 2007, pp. 259-263.
- [He08] Heise, D. et al.: Erweiterung einer Unternehmensmodellierungsmethode zur Unterstützung des IT-Controllings. In: Bichler, M. et al. (Eds.): Multikonferenz Wirtschaftsinformatik. Berlin 2008, pp. 1017-1028.
- [IT07] Office of Government Commerce (OGC): ITIL Version 3. The Stationery Office, Norwich, 2007.
- [Ju08] Junginger, S. et. al.: Anwendungsportfoliomanagement mit ADOit im ZIVIT. In: Riempp, G.; Stahringer, S. (Eds.): HDM-Praxis der Wirtschaftsinformatik: Unternehmensarchitekturen. dpunkt-verlag GmbH, 262, 2008, pp. 29-38.
- [Ke07] Keller, W.: IT-Unternehmensarchitektur. Von der Geschäftsstrategie zur optimalen IT Unterstützung. dpunkt.verlag, Heidelberg, 2007.
- [LS07] Lanka, J.; Schweda, C.: Constructing Application Landscape Metrics – Why and How. Technische Universität München, Institut für Informatik, Lehrstuhl für Informatik 19, Technischer Bericht TB0701, 2007.
- [MC07] Mohagheghi, P., Conradi, R.: Quality, productivity and economic benefits of software reuse: a review of industrial studies, Empirical Software Engineering, Vol. 12, No. 5, Oct 2007, pp. 471-516.
- [Mo09] Moser, C. et al.: Some Process Patterns for Enterprise Architecture Management. In: SE 2009 – Workshopband "Patterns in Enterprise Architecture Management (PEAM) 2009", Software Engineering 2009.
- [Mu08] Murer, S. et. al.: Managed Evolution – Nachhaltige Entwicklung großer Systeme. In: Informatik Spektrum. Vol. 31, No. 6, 2008, pp. 537-547.
- [RG07] Riempp, G.; Gieffers-Ankel, S.: Application portfolio management: a decision-oriented view of enterprise architecture. In: Information Systems and E-Business Management, Vol. 5, 2007, No. 4; pp. 359-378.
- [RW06] Ross, J. W., Weill, P., Robertson, D. C.: Enterprise Architecture as Strategy. Harvard Business School Press, Boston, 2006
- [Sc06] Schekkerman, J.: How to Survive in the Jungle of Enterprise Architecture Frameworks: Creating or Choosing an EA Framework. Trafford Publishing, 2006.
- [TO09] The Open Group Architecture Framework, <http://www.opengroup.org/togaf> (access 2009-05-15).
- [Va08] Vasconcelos, A. et. al: Enterprise Architecture Analysis – An Information System Evaluation Approach. In: Enterprise Modelling and Information Systems Architectures. Vol. 3, No. 2, December 2008, pp. 31-53.
- [Za65] Zadeh, L. A.: Fuzzy sets. In: Information and Control, Vol. 8, No. 3, June 1965, pp. 338-353.
- [ZI09] ZIVIT Homepage, <http://www.zivit.de> (access 2009-04-25).

GI-Edition Lecture Notes in Informatics

- P-1 Gregor Engels, Andreas Oberweis, Albert Zündorf (Hrsg.): Modellierung 2001.
- P-2 Mikhail Godlevsky, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications, ISTA'2001.
- P-3 Ana M. Moreno, Reind P. van de Riet (Hrsg.): Applications of Natural Language to Information Systems, NLDB'2001.
- P-4 H. Wörn, J. Mühling, C. Vahl, H.-P. Meinzer (Hrsg.): Rechner- und sensor-gestützte Chirurgie; Workshop des SFB 414.
- P-5 Andy Schürr (Hg.): OMER – Object-Oriented Modeling of Embedded Real-Time Systems.
- P-6 Hans-Jürgen Appelrath, Rolf Beyer, Uwe Marquardt, Heinrich C. Mayr, Claudia Steinberger (Hrsg.): Unternehmen Hochschule, UH'2001.
- P-7 Andy Evans, Robert France, Ana Moreira, Bernhard Rumpe (Hrsg.): Practical UML-Based Rigorous Development Methods – Countering or Integrating the extremists, pUML'2001.
- P-8 Reinhard Keil-Slawik, Johannes Magenheimer (Hrsg.): Informatikunterricht und Medienbildung, INFOS'2001.
- P-9 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Innovative Anwendungen in Kommunikationsnetzen, 15. DFN Arbeitstagung.
- P-10 Mirjam Minor, Steffen Staab (Hrsg.): 1st German Workshop on Experience Management: Sharing Experiences about the Sharing Experience.
- P-11 Michael Weber, Frank Kargl (Hrsg.): Mobile Ad-Hoc Netzwerke, WMAN 2002.
- P-12 Martin Glinz, Günther Müller-Luschnat (Hrsg.): Modellierung 2002.
- P-13 Jan von Knop, Peter Schirmbacher and Viljan Mahni_ (Hrsg.): The Changing Universities – The Role of Technology.
- P-14 Robert Tolksdorf, Rainer Eckstein (Hrsg.): XML-Technologien für das Semantic Web – XSW 2002.
- P-15 Hans-Bernd Bludau, Andreas Koop (Hrsg.): Mobile Computing in Medicine.
- P-16 J. Felix Hampe, Gerhard Schwabe (Hrsg.): Mobile and Collaborative Business 2002.
- P-17 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Zukunft der Netze –Die Verletzbarkeit meistern, 16. DFN Arbeitstagung.
- P-18 Elmar J. Sinz, Markus Plaha (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2002.
- P-19 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3.Okt. 2002 in Dortmund.
- P-20 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3.Okt. 2002 in Dortmund (Ergänzungsband).
- P-21 Jörg Desel, Mathias Weske (Hrsg.): Promise 2002: Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen.
- P-22 Sigrid Schubert, Johannes Magenheimer, Peter Hubwieser, Torsten Brinda (Hrsg.): Forschungsbeiträge zur “Didaktik der Informatik” – Theorie, Praxis, Evaluation.
- P-23 Thorsten Spitta, Jens Borchers, Harry M. Sneed (Hrsg.): Software Management 2002 – Fortschritt durch Beständigkeit
- P-24 Rainer Eckstein, Robert Tolksdorf (Hrsg.): XMIDX 2003 – XML-Technologien für Middleware – Middleware für XML-Anwendungen
- P-25 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Commerce – Anwendungen und Perspektiven – 3. Workshop Mobile Commerce, Universität Augsburg, 04.02.2003
- P-26 Gerhard Weikum, Harald Schöning, Erhard Rahm (Hrsg.): BTW 2003: Datenbanksysteme für Business, Technologie und Web
- P-27 Michael Kroll, Hans-Gerd Lipinski, Kay Melzer (Hrsg.): Mobiles Computing in der Medizin
- P-28 Ulrich Reimer, Andreas Abecker, Steffen Staab, Gerd Stumme (Hrsg.): WM 2003: Professionelles Wissensmanagement – Erfahrungen und Visionen
- P-29 Antje Düsterhöft, Bernhard Thalheim (Eds.): NLDB'2003: Natural Language Processing and Information Systems
- P-30 Mikhail Godlevsky, Stephen Liddle, Heinrich C. Mayr (Eds.): Information Systems Technology and its Applications
- P-31 Arslan Brömme, Christoph Busch (Eds.): BIOSIG 2003: Biometric and Electronic Signatures

- P-32 Peter Hubwieser (Hrsg.): Informatische Fachkonzepte im Unterricht – INFOS 2003
- P-33 Andreas Geyer-Schulz, Alfred Taudes (Hrsg.): Informationswirtschaft: Ein Sektor mit Zukunft
- P-34 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenber, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 1)
- P-35 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenber, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 2)
- P-36 Rüdiger Grimm, Hubert B. Keller, Kai Rannenber (Hrsg.): Informatik 2003 – Mit Sicherheit Informatik
- P-37 Arndt Bode, Jörg Desel, Sabine Rathmayer, Martin Wessner (Hrsg.): DeLFI 2003: e-Learning Fachtagung Informatik
- P-38 E.J. Sinz, M. Plaha, P. Neckel (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2003
- P-39 Jens Nedon, Sandra Frings, Oliver Göbel (Hrsg.): IT-Incident Management & IT-Forensics – IMF 2003
- P-40 Michael Rebstock (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2004
- P-41 Uwe Brinkschulte, Jürgen Becker, Dietmar Fey, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle, Thomas Runkler (Edts.): ARCS 2004 – Organic and Pervasive Computing
- P-42 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Economy – Transaktionen und Prozesse, Anwendungen und Dienste
- P-43 Birgitta König-Ries, Michael Klein, Philipp Obreiter (Hrsg.): Persistence, Scalability, Transactions – Database Mechanisms for Mobile Applications
- P-44 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): Security, E-Learning, E-Services
- P-45 Bernhard Rumpe, Wolfgang Hesse (Hrsg.): Modellierung 2004
- P-46 Ulrich Flegel, Michael Meier (Hrsg.): Detection of Intrusions of Malware & Vulnerability Assessment
- P-47 Alexander Prosser, Robert Krimmer (Hrsg.): Electronic Voting in Europe – Technology, Law, Politics and Society
- P-48 Anatoly Doroshenko, Terry Halpin, Stephen W. Liddle, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications
- P-49 G. Schiefer, P. Wagner, M. Morgenstern, U. Rickert (Hrsg.): Integration und Datensicherheit – Anforderungen, Konflikte und Perspektiven
- P-50 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 1) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm
- P-51 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 2) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm
- P-52 Gregor Engels, Silke Seehusen (Hrsg.): DELFI 2004 – Tagungsband der 2. e-Learning Fachtagung Informatik
- P-53 Robert Giegerich, Jens Stoye (Hrsg.): German Conference on Bioinformatics – GCB 2004
- P-54 Jens Borchers, Ralf Kneuper (Hrsg.): Softwaremanagement 2004 – Outsourcing und Integration
- P-55 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): E-Science und Grid Ad-hoc-Netze Medienintegration
- P-56 Fernand Feltz, Andreas Oberweis, Benoit Otjacques (Hrsg.): EMISA 2004 – Informationssysteme im E-Business und E-Government
- P-57 Klaus Turowski (Hrsg.): Architekturen, Komponenten, Anwendungen
- P-58 Sami Beydeda, Volker Gruhn, Johannes Mayer, Ralf Reussner, Franz Schweiggert (Hrsg.): Testing of Component-Based Systems and Software Quality
- P-59 J. Felix Hampe, Franz Lehner, Key Pousttchi, Kai Rannenber, Klaus Turowski (Hrsg.): Mobile Business – Processes, Platforms, Payments
- P-60 Steffen Friedrich (Hrsg.): Unterrichtskonzepte für informatische Bildung
- P-61 Paul Müller, Reinhard Gotzhein, Jens B. Schmitt (Hrsg.): Kommunikation in verteilten Systemen
- P-62 Federrath, Hannes (Hrsg.): „Sicherheit 2005“ – Sicherheit – Schutz und Zuverlässigkeit
- P-63 Roland Kaschek, Heinrich C. Mayr, Stephen Liddle (Hrsg.): Information Systems – Technology and its Applications

- P-64 Peter Liggesmeyer, Klaus Pohl, Michael Goedicke (Hrsg.): Software Engineering 2005
- P-65 Gottfried Vossen, Frank Leymann, Peter Lockemann, Wolffried Stucky (Hrsg.): Datenbanksysteme in Business, Technologie und Web
- P-66 Jörg M. Haake, Ulrike Lucke, Djamshid Tavangarian (Hrsg.): DeLFI 2005: 3. deutsche e-Learning Fachtagung Informatik
- P-67 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 1)
- P-68 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 2)
- P-69 Robert Hirschfeld, Ryszard Kowalczyk, Andreas Polze, Matthias Weske (Hrsg.): NODe 2005, GSEM 2005
- P-70 Klaus Turowski, Johannes-Maria Zaha (Hrsg.): Component-oriented Enterprise Application (COAE 2005)
- P-71 Andrew Torda, Stefan Kurz, Matthias Rarey (Hrsg.): German Conference on Bioinformatics 2005
- P-72 Klaus P. Jantke, Klaus-Peter Fähnrich, Wolfgang S. Wittig (Hrsg.): Marktplatz Internet: Von e-Learning bis e-Payment
- P-73 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): "Heute schon das Morgen sehen"
- P-74 Christopher Wolf, Stefan Lucks, Po-Wah Yau (Hrsg.): WEWoRC 2005 – Western European Workshop on Research in Cryptology
- P-75 Jörg Desel, Ulrich Frank (Hrsg.): Enterprise Modelling and Information Systems Architecture
- P-76 Thomas Kirste, Birgitta König-Riess, Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Informationssysteme – Potentiale, Hindernisse, Einsatz
- P-77 Jana Dittmann (Hrsg.): SICHERHEIT 2006
- P-78 K.-O. Wenkel, P. Wagner, M. Morgens-tern, K. Luzi, P. Eisermann (Hrsg.): Land- und Ernährungswirtschaft im Wandel
- P-79 Bettina Biel, Matthias Book, Volker Gruhn (Hrsg.): Softwareengineering 2006
- P-80 Mareike Schoop, Christian Huemer, Michael Rebstock, Martin Bichler (Hrsg.): Service-Oriented Electronic Commerce
- P-81 Wolfgang Karl, Jürgen Becker, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle (Hrsg.): ARCS '06
- P-82 Heinrich C. Mayr, Ruth Breu (Hrsg.): Modellierung 2006
- P-83 Daniel Huson, Oliver Kohlbacher, Andrei Lupas, Kay Nieselt and Andreas Zell (eds.): German Conference on Bioinformatics
- P-84 Dimitris Karagiannis, Heinrich C. Mayr, (Hrsg.): Information Systems Technology and its Applications
- P-85 Witold Abramowicz, Heinrich C. Mayr, (Hrsg.): Business Information Systems
- P-86 Robert Krimmer (Ed.): Electronic Voting 2006
- P-87 Max Mühlhäuser, Guido Röbling, Ralf Steinmetz (Hrsg.): DELFI 2006: 4. e-Learning Fachtagung Informatik
- P-88 Robert Hirschfeld, Andreas Polze, Ryszard Kowalczyk (Hrsg.): NODe 2006, GSEM 2006
- P-90 Joachim Schelp, Robert Winter, Ulrich Frank, Bodo Rieger, Klaus Turowski (Hrsg.): Integration, Informationslogistik und Architektur
- P-91 Henrik Stormer, Andreas Meier, Michael Schumacher (Eds.): European Conference on eHealth 2006
- P-92 Fernand Feltz, Benoît Otjacques, Andreas Oberweis, Nicolas Poussing (Eds.): AIM 2006
- P-93 Christian Hochberger, Rüdiger Liskowsky (Eds.): INFORMATIK 2006 – Informatik für Menschen, Band 1
- P-94 Christian Hochberger, Rüdiger Liskowsky (Eds.): INFORMATIK 2006 – Informatik für Menschen, Band 2
- P-95 Matthias Weske, Markus Nüttgens (Eds.): EMISA 2005: Methoden, Konzepte und Technologien für die Entwicklung von dienstbasierten Informationssystemen
- P-96 Saartje Brockmans, Jürgen Jung, York Sure (Eds.): Meta-Modelling and Ontologies
- P-97 Oliver Göbel, Dirk Schadt, Sandra Frings, Hardo Hase, Detlef Günther, Jens Nedon (Eds.): IT-Incident Mangament & IT-Forensics – IMF 2006

- P-98 Hans Brandt-Pook, Werner Simonsmeier und Thorsten Spitta (Hrsg.): Beratung in der Softwareentwicklung – Modelle, Methoden, Best Practices
- P-99 Andreas Schwill, Carsten Schulte, Marco Thomas (Hrsg.): Didaktik der Informatik
- P-100 Peter Forbrig, Günter Siegel, Markus Schneider (Hrsg.): HDI 2006: Hochschuldidaktik der Informatik
- P-101 Stefan Böttinger, Ludwig Theuvsen, Susanne Rank, Marlies Morgenstern (Hrsg.): Agrarinformatik im Spannungsfeld zwischen Regionalisierung und globalen Wertschöpfungsketten
- P-102 Otto Spaniol (Eds.): Mobile Services and Personalized Environments
- P-103 Alfons Kemper, Harald Schöning, Thomas Rose, Matthias Jarke, Thomas Seidl, Christoph Quix, Christoph Brochhaus (Hrsg.): Datenbanksysteme in Business, Technologie und Web (BTW 2007)
- P-104 Birgitta König-Ries, Franz Lehner, Rainer Malaka, Can Türker (Hrsg.): MMS 2007: Mobilität und mobile Informationssysteme
- P-105 Wolf-Gideon Bleek, Jörg Raasch, Heinz Züllighoven (Hrsg.): Software Engineering 2007
- P-106 Wolf-Gideon Bleek, Henning Schwentner, Heinz Züllighoven (Hrsg.): Software Engineering 2007 – Beiträge zu den Workshops
- P-107 Heinrich C. Mayr, Dimitris Karagiannis (eds.) Information Systems Technology and its Applications
- P-108 Arslan Brömme, Christoph Busch, Detlef Hühnlein (eds.) BIOSIG 2007: Biometrics and Electronic Signatures
- P-109 Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.): INFORMATIK 2007 Informatik trifft Logistik Band 1
- P-110 Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.): INFORMATIK 2007 Informatik trifft Logistik Band 2
- P-111 Christian Eibl, Johannes Magenheimer, Sigrid Schubert, Martin Wessner (Hrsg.): DeLFI 2007: 5. e-Learning Fachtagung Informatik
- P-112 Sigrid Schubert (Hrsg.) Didaktik der Informatik in Theorie und Praxis
- P-113 Sören Auer, Christian Bizer, Claudia Müller, Anna V. Zhdanova (Eds.) The Social Semantic Web 2007 Proceedings of the 1st Conference on Social Semantic Web (CSSW)
- P-114 Sandra Frings, Oliver Göbel, Detlef Günther, Hardo G. Hase, Jens Nedon, Dirk Schadt, Arslan Brömme (Eds.) IMF2007 IT-incident management & IT-forensics Proceedings of the 3rd International Conference on IT-Incident Management & IT-Forensics
- P-115 Claudia Falter, Alexander Schliep, Joachim Selbig, Martin Vingron and Dirk Walther (Eds.) German conference on bioinformatics GCB 2007
- P-116 Witold Abramowicz, Leszek Maciszek (Eds.) Business Process and Services Computing 1st International Working Conference on Business Process and Services Computing BPSC 2007
- P-117 Ryszard Kowalczyk (Ed.) Grid service engineering and management The 4th International Conference on Grid Service Engineering and Management GSEM 2007
- P-118 Andreas Hein, Wilfried Thoben, Hans-Jürgen Appelrath, Peter Jensch (Eds.) European Conference on ehealth 2007
- P-119 Manfred Reichert, Stefan Strecker, Klaus Turowski (Eds.) Enterprise Modelling and Information Systems Architectures Concepts and Applications
- P-120 Adam Pawlak, Kurt Sandkuhl, Wojciech Cholewa, Leandro Soares Indrusiak (Eds.) Coordination of Collaborative Engineering - State of the Art and Future Challenges
- P-121 Korbinian Herrmann, Bernd Bruegge (Hrsg.) Software Engineering 2008 Fachtagung des GI-Fachbereichs Softwaretechnik
- P-122 Walid Maalej, Bernd Bruegge (Hrsg.) Software Engineering 2008 - Workshopband Fachtagung des GI-Fachbereichs Softwaretechnik

- P-123 Michael H. Breitner, Martin Breunig, Elgar Fleisch, Ley Pousttchi, Klaus Turowski (Hrsg.)
Mobile und Ubiquitäre Informationssysteme – Technologien, Prozesse, Marktfähigkeit
Proceedings zur 3. Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2008)
- P-124 Wolfgang E. Nagel, Rolf Hoffmann, Andreas Koch (Eds.)
9th Workshop on Parallel Systems and Algorithms (PASA)
Workshop of the GI/ITG Special Interest Groups PARS and PARVA
- P-125 Rolf A.E. Müller, Hans-H. Sundermeier, Ludwig Theuvsen, Stephanie Schütze, Marlies Morgenstern (Hrsg.)
Unternehmens-IT:
Führungsinstrument oder Verwaltungsbürde
Referate der 28. GIL Jahrestagung
- P-126 Rainer Gimnich, Uwe Kaiser, Jochen Quante, Andreas Winter (Hrsg.)
10th Workshop Software Reengineering (WSR 2008)
- P-127 Thomas Kühne, Wolfgang Reisig, Friedrich Steimann (Hrsg.)
Modellierung 2008
- P-128 Ammar Alkassar, Jörg Siekmann (Hrsg.)
Sicherheit 2008
Sicherheit, Schutz und Zuverlässigkeit
Beiträge der 4. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)
2.-4. April 2008
Saarbrücken, Germany
- P-129 Wolfgang Hesse, Andreas Oberweis (Eds.)
Sigsand-Europe 2008
Proceedings of the Third AIS SIGSAND European Symposium on Analysis, Design, Use and Societal Impact of Information Systems
- P-130 Paul Müller, Bernhard Neumair, Gabi Dreö Rodosek (Hrsg.)
1. DFN-Forum Kommunikationstechnologien Beiträge der Fachtagung
- P-131 Robert Krimmer, Rüdiger Grimm (Eds.)
3rd International Conference on Electronic Voting 2008
Co-organized by Council of Europe, Gesellschaft für Informatik and E-Voting.CC
- P-132 Silke Seehusen, Ulrike Lucke, Stefan Fischer (Hrsg.)
DeLFI 2008:
Die 6. e-Learning Fachtagung Informatik
- P-133 Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, Christian Scheideler (Hrsg.)
INFORMATIK 2008
Beherrschbare Systeme – dank Informatik Band 1
- P-134 Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, Christian Scheideler (Hrsg.)
INFORMATIK 2008
Beherrschbare Systeme – dank Informatik Band 2
- P-135 Torsten Brinda, Michael Fothe, Peter Hubwieser, Kirsten Schlüter (Hrsg.)
Didaktik der Informatik – Aktuelle Forschungsergebnisse
- P-136 Andreas Beyer, Michael Schroeder (Eds.)
German Conference on Bioinformatics GCB 2008
- P-137 Arslan Brömme, Christoph Busch, Detlef Hühnlein (Eds.)
BIOSIG 2008: Biometrics and Electronic Signatures
- P-138 Barbara Dinter, Robert Winter, Peter Chamoni, Norbert Gronau, Klaus Turowski (Hrsg.)
Synergien durch Integration und Informationslogistik
Proceedings zur DW2008
- P-139 Georg Herzwurm, Martin Mikusz (Hrsg.)
Industrialisierung des Software-Managements
Fachtagung des GI-Fachausschusses Management der Anwendungsentwicklung und -wartung im Fachbereich Wirtschaftsinformatik
- P-140 Oliver Göbel, Sandra Frings, Detlef Günther, Jens Nedon, Dirk Schadt (Eds.)
IMF 2008 - IT Incident Management & IT Forensics
- P-141 Peter Loos, Markus Nüttgens, Klaus Turowski, Dirk Werth (Hrsg.)
Modellierung betrieblicher Informationssysteme (MobIS 2008)
Modellierung zwischen SOA und Compliance Management
- P-142 R. Bill, P. Korduan, L. Theuvsen, M. Morgenstern (Hrsg.)
Anforderungen an die Agrarinformatik durch Globalisierung und Klimaveränderung
- P-143 Peter Liggesmeyer, Gregor Engels, Jürgen Münch, Jörg Dörr, Norman Riegel (Hrsg.)
Software Engineering 2009
Fachtagung des GI-Fachbereichs Softwaretechnik

- P-144 Johann-Christoph Freytag, Thomas Ruf,
Wolfgang Lehner, Gottfried Vossen
(Hrsg.)
Datenbanksysteme in Business,
Technologie und Web (BTW)
- P-145 Knut Hinkelmann, Holger Wache (Eds.)
WM2009: 5th Conference on Professional
Knowledge Management
- P-146 Markus Bick, Martin Breunig,
Hagen Höpfner (Hrsg.)
Mobile und Ubiquitäre
Informationssysteme – Entwicklung,
Implementierung und Anwendung
4. Konferenz Mobile und Ubiquitäre
Informationssysteme (MMS 2009)
- P-147 Witold Abramowicz, Leszek Maciaszek,
Ryszard Kowalczyk, Andreas Speck (Eds.)
Business Process, Services Computing
and Intelligent Service Management
BPSC 2009 · ISM 2009 · YRW-MBP 2009
- P-148 Christian Erfurth, Gerald Eichler,
Volkmar Schau (Eds.)
9th International Conference on Innovative
Internet Community Systems
I²CS 2009
- P-149 Paul Müller, Bernhard Neumair,
Gabi Dreo Rodosek (Hrsg.)
2. DFN-Forum
Kommunikationstechnologien
Beiträge der Fachtagung
- P-150 Jürgen Münch, Peter Liggesmeyer (Hrsg.)
Software Engineering
2009 - Workshopband
- P-151 Armin Heinzl, Peter Dadam, Stefan Kirn,
Peter Lockemann (Eds.)
PRIMIUM
Process Innovation for
Enterprise Software
- P-152 Jan Mendling, Stefanie Rinderle-Ma,
Werner Esswein (Eds.)
Enterprise Modelling and Information
Systems Architectures
Proceedings of the 3rd Int'l Workshop
EMISA 2009
- P-153 Andreas Schwill,
Nicolas Apostolopoulos (Hrsg.)
Lernen im Digitalen Zeitalter
DeLFI 2009 – Die 7. E-Learning
Fachtagung Informatik

The titles can be purchased at:

Köllen Druck + Verlag GmbH

Ernst-Robert-Curtius-Str. 14 · D-53117 Bonn

Fax: +49 (0)228/9898222

E-Mail: druckverlag@koellen.de