Ilia Bider Terry Halpin
John Krogstie Selmin Nurcan
Erik Proper Rainer Schmidt
Roland Ukor (Eds.)

LNBIP 50

Enterprise, Business-Process and Information Systems Modeling

11th International Workshop, BPMDS 2010
and 15th International Conference, EMMSAD 2010
held at CAiSE 2010, Hammamet, Tunisia, June 2010
Proceedings

Springer

Lecture Notes
in Business Information Processing 50

Ilia Bider Terry Halpin John Krogstie
Selmin Nurcan Erik Proper
Rainer Schmidt Roland Ukor (Eds.)

Enterprise, Business-Process and Information Systems Modeling

11th International Workshop, BPMDS 2010
and 15th International Conference, EMMSAD 2010
held at CAiSE 2010, Hammamet, Tunisia, June 7-8, 2010
Proceedings

Springer

Volume Editors

Ilia Bider
IbisSoft AB, Stockholm, Sweden
E-mail: ilia@ibissoft.se

Terry Halpin
LogicBlox, Australia, and INTI Education Group, Malaysia
E-mail: terry.halpin@logicblox.com

John Krogstie
Norwegian University of Science and Technology, NTNU, Trondheim, Norway
E-mail: john.krogstie@idi.ntnu.no

Selmin Nurcan
University of Paris 1 Pantheon Sorbonne, Paris, France
E-mail: selmin.nurcan@univ-paris.fr

Erik Proper
Public Research Centre Henri Tudor, Luxembourg
and Radboud University Nijmegen, The Netherlands
E-mail: e.proper@acm.org

Rainer Schmidt
Aalen University, Aalen, Germany
E-mail: rainer.schmidt@htw-aalen.de

Roland Ukor
University of Manchester, UK
E-mail: roland.ukor@cs.man.ac.uk

Library of Congress Control Number: 2010926753

ACM Computing Classification (1998): J.1, H.4.1, H.3.5, D.2

ISSN 1865-1348
ISBN-10 3-642-13050-X Springer Berlin Heidelberg New York
ISBN-13 978-3-642-13050-2 Springer Berlin Heidelberg New York

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

Preface

This book contains the proceedings of two well established scientific events held in connection with the CAiSE conferences relating to the areas of enterprise, business-processes, and information systems modeling:

- The 11th International Workshop on Business Process Modeling, Development and Support (BPMDS 2010);
- The 15th International Conference on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD 2010).

The two events are introduced briefly below.

BPMDS 2010

BPMDS 2010 was the 11th in a series of workshops that have successfully served as a forum for raising and discussing new ideas in the area of business process development and support.

The BPMDS series has produced 10 workshops from 1998 to 2009. Eight of these workshops, including the last seven (BPMDS 2003–BPMDS 2009) were held in conjunction with CAiSE conferences. The BPMDS workshops focus on topics relating to IT support for business processes, which addresses key issues that are relevant to the continuous development of information systems theory. The continued interest in these topics within the industrial and academic IS communities is reflected by the success of the last BPMDS workshops and the emergence of new conferences devoted to this theme.

Previous BPMDS workshops focused on the different phases in the business process life-cycle as well as the drivers that motivate and initiate business process design and evolution.

This edition of the BPMDS workshop (BPMDS 2010) focused on non-dominant perspectives of business processes and their integration. The workshop was devoted to the following three topics:

- Non-dominant perspectives/sets of perspectives on business processes; for instance, goal, state, context, and resource;
- Finding out which perspectives are most appropriate to particular practical and/or theoretical tasks of business process modeling, development and support (BPMDS); finding BPMDS tasks/problems that can be accomplished/solved when using a particular perspective;
- Connecting several perspectives (including a dominant one) together. Each perspective can be considered as a projection of the business process in a particular dimension and a set of perspectives can be considered as a way of organizing a multi-dimensional space. Therefore, connecting several perspectives can provide a multi-dimensional representation of the business process.

The 13 papers accepted to BPMDS 2010 were selected from among 27 papers submitted from 17 countries (Algeria, Australia, Austria, Cameroun, China, France, Germany, Israel, Japan, Jordan, Luxembourg, Morocco, Spain, Sweden, The Netherlands, Tunisia and Turkey). They cover a wide spectrum of issues related to the multi-dimensional perspectives on business processes. They are organized under the following section headings:

- State oriented perspective
- Service provision as a perspective
- Knowledge sharing and collaboration as perspectives
- 'Fine-tuning' as a perspective: scheduling, configuration and efficiency
- Integrating multiple perspectives

We wish to thank all the people who submitted papers to the workshop for having shared their work with us, as well as the members of the BPMDS 2010 Program Committee and the workshop organizers of CAiSE 2010 for their help with the organization of the workshop.

The goals, format, and history of BPMDS can be found on the web site: http://www.ibissoft.se/bpmds.html

March 2010

Ilia Bider
Selmin Nurcan
Rainer Schmidt
Roland Ukor

Organization

BPMDS 2010 Industrial Advisory Board

Ian Alexander	Scenario Plus, UK
Ilia Bider	Ibisoft, Sweden
Gil Regev	EPFL and Itecor, Switzerland
Lars Taxén	Linköping University, Sweden

BPMDS 2010 Organizing Committee

Ilia Bider	Ibisoft, Sweden
Selmin Nurcan	University Paris 1 Pantheon Sorbonne, France
Rainer Schmidt	Aelen University of Applied Sciences, Germany
Roland Ukor	University of Manchester, UK

BPMDS 2010 Program Committee

Wil van der Aalst	Eindhoven University of Technology, The Netherlands
Sebastian Adam	Fraunhofer IESE, Kaiserslautern, Germany
Antonia Albani	Delft University of Technology, The Netherlands
Ian Alexander	Scenario Plus, UK
Ilia Bider	IbisSoft, Stockholm, Sweden
Stewart Green	University of the West of England, UK
Paul Johannesson	Royal University of Technology, Stockholm, Sweden
Agnes Koschmider	Karlsruhe Institute of Technology, Germany
Marite Kirikova	Riga Technical University, Latvia
Peri Loucopoulos	Loughborough University, UK
Renata Mendes de Araujo	Federal University of the State of Rio de Janeiro, Brazil
Jan Mendling	Humboldt University of Berlin, Germany
Selmin Nurcan	University Paris 1 Pantheon Sorbonne, France
Louis-François Pau	Erasmus University, The Netherlands
Jan Recker	Queensland University of Technology, Brisbane, Australia
Gil Regev	EPFL and Itecor, Switzerland

Manfred Reichert	University of Ulm, Germany
Michael Rosemann	Queensland University of Technology, Brisbane, Australia
Rainer Schmidt	University of Applied Sciences, Aalen, Germany
Pnina Soffer	University of Haifa, Israel
Markus Strohmaier	University of Toronto, Canada
Lars Taxén	Linköping University, Sweden
Roland Ukor	University of Manchester, UK
Barbara Weber	University of Innsbruck, Austria
Jelena Zdravkovic	Royal University of Technology, Stockholm, Sweden

EMMSAD 2010

The field of information systems analysis and design includes numerous information modeling methods and notations (e.g., ER, ORM, UML, DFDs, BPMN), that are typically evolving. Even with some attempts to standardize (e.g., UML for object-oriented design), new modeling methods are constantly being introduced, many of which differ only marginally from existing approaches. These ongoing changes significantly impact the way information systems are analyzed and designed in practice. This conference focuses on exploring, evaluating, and enhancing current information modeling methods and methodologies. Though the need for such studies is well recognized, there is a paucity of such research in the literature.

The objective of EMMSAD 2010 was to provide a forum for researchers and practitioners interested in modeling methods in systems analysis and design to meet, and exchange research ideas and results. It also gave the participants an opportunity to present their research papers and experience reports, and to take part in open discussions.

EMMSAD 2010 was the fifteenth in a very successful series of EMMSAD events, previously held in Heraklion, Barcelona, Pisa, Heidelberg, Stockholm, Interlaken, Toronto, Velden, Riga, Porto, Luxembourg, Trondheim, Montpellier and Amsterdam. This year we had 22 papers submitted by authors from 16 different countries (Australia, Brazil, France, Germany, Ireland, Israel, Italy, Luxembourg, Malaysia, The Netherlands, Norway, South Africa, Spain, Sweden, Switzerland and Tunisia). After an extensive review process by a distinguished international program committee, with each paper receiving at least three reviews, we accepted the 14 papers that appear in these proceedings. Congratulations to the successful authors!

Apart from the contributions of the authors, the quality of this conference depended in no small way on the generous contribution of time and effort by the program committee and the additional reviewers. Their work is greatly appreciated. Continuing with our very successful collaboration with IFIP WG 8.1 that started in 1997, this year's conference was again a joint activity of CAiSE and WG 8.1. The European INTEROP Network of Excellence has also sponsored this conference since 2005, as has AIS-SIGSAND. For more information on the conference, see our website `http://www.emmsad.org`.

March 2010

Erik Proper
John Krogstie
Terry Halpin

Organization

EMMSAD Steering Committee

Terry Halpin	LogicBlox, Australia & INTI University College, Malaysia
John Krogstie	Norwegian Institute of Science and Technology/SINTEF, Norway
Keng Siau	University of Nebraska-Lincoln, USA

Conference Co-chairs

Erik Proper	Public Research Centre Henri Tudor, Luxembourg & Radboud University Nijmegen, The Netherlands
John Krogstie	Norwegian Institute of Science and Technology/SINTEF, Norway
Terry Halpin	LogicBlox, Australia & INTI University College, Malaysia

Program Committee of EMMSAD 2010

Eric Dubois	Centre Henri Tudor, Luxembourg
Florian Matthes	Technical University Munich, Germany
Mathias Ekstedt	Royal Institute of Technology, Sweden
Pontus Johnson	Royal Institute of Technology, Sweden
Antonia Albani	University of St. Gallen, Switzerland
Sietse Overbeek	Delft University of Technology, The Netherlands
Bas van Gils	BiZZdesign, The Netherlands
Annie Becker	Florida Institute of Technology, USA
Giuseppe Berio	University of Turin, Italy
Nacer Boudjlida	UHP Nancy 1/Loria, France
Inge van de Weerd	Utrecht University, The Netherlands
Andy Carver	Carver Consulting, USA
Olga De Troyer	Vrije Universiteit Brussel, Belgium
John Erickson	University of Nebraska-Omaha, USA
Peter Fettke	Institute for Information Systems, DFKI, Germany
Ulrich Frank	University of Duisberg-Essen, Germany
Andrew Gemino	Simon Fraser University, Canada
Gøran Goldkuhl	Linkøping University, Sweden

Remigijus Gustas	Karlstad University, Sweden
Frank Harmsen	Ernst & Young and Maastricht University, The Netherlands
Wolfgang Hesse	Philipps-University Marburg, Germany
Stijn Hoppenbrouwers	Radboud University Nijmegen, The Netherlands
Jon Iden	Norges Handelshøyskole, Bergen, Norway
Paul Johanneson	Stockholm University, Sweden
Pericles Loucopoulos	Loughborough University, UK
Graham McLeod	Promis Solutions, Switzerland
Jan Mendling	Humboldt-Universität zu Berlin, Germany
Tony Morgan	Neumont University, USA
Michele Missikoff	LEKS, IASI, Italy
Andreas L. Opdahl	University of Bergen, Norway
Hervé Panetto	University Henri Poincaré Nancy I, France
Barbara Pernici	Politecnico di Milano, Italy
Anne Persson	University of Skövde, Sweden
Michaël Petit	University of Namur, Belgium
Jolita Ralyté	University of Geneva, Switzerland
Sudha Ram	University of Arizona, USA
Jan Recker	Queensland University of Technology, Australia
Colette Rolland	University of Paris 1, France
Michael Rosemann	Queensland University of Technology, Australia
Matti Rossi	Helsinki School of Economics, Finland
Kurt Sandkuhl	Jönköping University, Sweden
Peretz Shoval	Ben-Gurion University of the Negev, Israel
Il-Yeol Song	Drexel University, USA
Janis Stirna	Royal Institute of Technology, Sweden
Johan Versendaal	University of Utrecht, The Netherlands
Carson Woo	University of British Columbia, Canada
Martin Zelm	CIMOSA, Germany
Pär Ågerfalk	Uppsala University, Sweden

Additional Reviewers for EMMSAD 2010

Alexis Aubry
Markus Buschle
Ulrik Franke
Jens Gulden
Marijke Janssen
Heiko Kattenstroth
Robert Lagerström
Francesco Taglino
Esma Yahia

Table of Contents

BPMDS 2010

State-Oriented Perspective

Service Provision as a Perspective

Knowledge Sharing and Collaboration as Perspectives

'Fine-Tuning' as a Perspective: Scheduling, Configuration and Efficiency

Integrating Multiple Perspectives

EMMSAD 2010

In Search of the Holy Grail: Integrating Social Software with BPM Experience Report

Ilia Bider[1,2], Paul Johannesson[1], and Erik Perjons[1]

[1] DSV, Stockholm University, Stockholm, Forum 100, SE-16440 Kista, Sweden
[2] IbisSoft AB, Stockholm, Box 19567, SE-10432 Stockholm, Sweden
ilia@ibissoft.se, {pajo,perjons}@dsv.su.se

Abstract. The paper is devoted to finding a view on business processes that helps to introduce into business process support systems a notion of shared spaces widely used in social software. The paper presents and analyses the experience of the authors from a number of development projects aimed at building business process support systems. The authors define a role that shared spaces can play in business process support and set some requirements on the shared space structure based on this role. They then analyze their projects in order to show how these requirements can be met and describe what practical results have been achieved in each project.

Keywords: business process, information logistics, social software, state-oriented.

1 Introduction

One of today's trends is a growing use of social software, e.g. Facebook, Twitter and Flickr, in private life. A new generation is growing up that is accustomed to communicate with each other through social software. Through this generation, this new way of communication is quickly spreading to business life. Business-oriented sites, such as LinkedIn, are widely used for informal business networks, personal marketing and sales. The ideas built into social software has begun to affect the design of business-oriented software systems, including Business Process (BP) support systems, which is reflected in several new directions in contemporary IS research [1].

Social software is based on the idea of common spaces shared by many individuals. This kind of software is used mainly for ad-hoc communication. Business process support is a system that helps process participants to drive their processes in a structured way towards specific "operational" goals. The question is how to "marry" these two on the surface different worlds in order to attain the benefits of social software when running business processes using a BP support system. The majority of today's BP support systems are built upon the workflow view on BP. It is difficult to see how to add the idea of shared spaces to this view in a natural way. Another view on BP is needed for merging social software and business process support.

This paper presents an experience report that describes our efforts in finding a view on BP that allows integration of the idea of shared spaces from ad-hoc social

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 1–13, 2010.

communication with the goal-orientedness of business processes. Our underlying theory in this search was (and still is) the state-oriented view on business processes [2]. In this theory, a business process instance is defined as a trajectory in a state space, the driving forth of movement in which being people completing various activities (tasks). Activities are planned based on the position in the state space and sometimes on the process history.

Our initial practical experience has shown that a direct implementation of the ideas from [2] in a BP support system has a number of drawbacks. Consequently, some amendments to this view have been made to design BP support systems that both implement the idea of shared spaces and are convenient for end-users. The amended view has some similarity with the workflow view as it represents a process (type) as a number of boxes placed one after another (with the possibility to put some of the boxes upon others). The semantics behind the boxes is, however, different. Each box represents a subspace of the process state-space. Positioning of the boxes reflects the requirements on the instance trajectories, e.g., a progress in some subspaces is required before starting movement in some other subspaces.

Our experience report is structured in the following way. In section 2, we explain our view on a role of shared spaces in a system that supports BPs. In section 3, we formulate requirements on shared spaces for BP support. In section 4, we describe our initial experience of introducing shared spaces in BP support systems and explain why the initial plan has not been as successful as we hoped. In section 5, we describe how to build a BP support system based on the amended state-oriented view and discuss the advantages that it brings. Section 6 contains concluding remarks and plans for the future.

2 A Role of Shared Spaces in BP Support Systems

Most of the young students whom we asked the question on what is the best thing with social software gave a straightforward answer: communication possibilities. You publish something, e.g., a photo album, once and can then make it available to as many people as you like and when you like it by inviting them to visit your space. The answer comes as no surprise, as communication is what social software is designed for. If we turn our attention to BP support systems, communication is not a primary objective here. The focus is on reaching the goal set for a process instance with as little communication as possible in order to be efficient. To introduce shared spaces in BP support, we first need to find their possible role in running business processes.

Roughly, there are two types of communication in the frame of a process instance, communicating the assignment of tasks and communicating information needed for completing the tasks. Communicating assignments is not exceptionally difficult, and most BP support systems handle it satisfactory. Communicating information needed for the task execution represents a bigger problem, as it is not always possible to know beforehand how much information may be needed for completing a task in a particular process instance.

In this paper, we focus on the second type of communication – communicating information needed for the task execution. In fact, on a more abstract level, the goal

here is not communication, but providing information. In analogy with the physical world, we can employ the concept of information logistics when dealing with this second type of communication in the frame of a process instance.

The term *information logistics* is relatively new. According to Wikipedia (which does not contradict with other sources on the matter), information logistics means: "providing the right information to the right recipients at the right time and place". We do not fully agree with this definition, as, in our view, it implies "moving information to recipients". This is in collision with the Requirement Engineering rule of maximum separation between the problem and solution domains. A better definition of information logistics for our purposes would sound like "bringing together information and people (or other type of agents, e.g. machines) who should process this information in the frame of a business process (instance)". This is a more neutral definition, as it allows various logistics schemes, such as:

- Moving information to people
- Moving people to information
- Or any combination of the first two

Let us consider an analogy with production processes. In a production process, the term logistic means "bringing together physical objects and people (or other agents) who will complete some operations on them". In production, both a scheme when objects are moved to people, a.k.a. conveyor belt logistics, and a scheme when people are moved to objects, a.k.a., construction site logistics, are widely used, see Fig. 1.

Fig. 1. Two types of logistics in production processes: to the left - conveyor belt logistics, to the right - construction site logistic

As far as business processes are concerned, the predominant way of arranging information logistics was, and still is, the conveyor belt scheme. This is quite understandable, because:

- Moving information, e.g. via mail, has been much cheaper than moving people
- Arranging people movement in an office would be a challenging task, see Fig. 2.

Introducing a BP support system moves us from the physical world to the virtual one. Movements in the virtual world do not cost much, and they are easy to arrange.

For example, it is easy to move between different on-line bookstores, and people do not run into one another while doing so. Therefore, the costs and difficulties of arranging people movements are no more reasons to prefer the conveyor belt information logistics rather than the construction site one when designing business process support.

In addition, let us look at the production analogy more attentively. The conveyor belt logistics is extremely efficient for producing the same kind of goods over and over again, but nobody sets a conveyor belt if one needs to produce a personal car, a bus, and a lorry at random. This is exactly the kind of situations for many business processes; one instance can be as different from another as a personal car from a lorry. Thus, the efficiency of the conveyor belt in production may not be easily translated to its efficiency for business processes. At least, there is no logical reason for such an assumption.

Fig. 2. Moving people to information in a physical world is a challenging task

When there are substantial differences between the instances of a business process, it is difficult to decide what and how much information needs to be sent to a person completing a certain task. Choosing a construction site logistic here has an advantage. If we move a worker to a certain place inside a construction site, he oversees not only this local place, but also everything adjacent to it, and can use this information, if necessary, when completing his/her task without being explicitly told to do so. The same is true for business processes. When you send just one document to a person, this is all he/she gets. If you send a person to work on a certain document placed in some corner of a desk (see Fig. 2), he/she can access not only this document, but also other documents in this corner, or on the whole desk.

The above deliberations (more on this see [3]) lead us to the conclusion that the construction site information logistics can be preferable for business processes that are supported by a software system. A shared space in such a system plays a role of a construction site: it holds all information that is relevant to a process instance, e.g., document received and sent, information on tasks planned and completed, reports on results achieved when completing these tasks, etc. All this information is easily available each time a process participant is invited to visit this space and complete some task related to it.

3 Requirements on Shared Spaces for BP Support

The functioning of a BP support system based on shared spaces can be described in the following way. Let us assume that the system has a capacity of creating any number of shared spaces, then:

- When a new process instance/case starts, a new shared space is created. It gets a unique name, an owner (responsible for the case), and possibly, a case team.
- When the process instance reaches its operational goal, the shared space is closed (sealed), but remains accessible for reading (a case goes to the archive).
- A person who is assigned a task in the frame of the process case "goes" to the shared space of the case in order to get the information he needs for completing the task and reports the results achieved in the same space.
- A shared space can be public, private, or restricted. If public, any worker can pop up in the "unlocked space" to see what is going on, and leave some traces of his visit, e.g. personal comments. If private, only the owner and members of the case team have access to the space. Restricted means the access is controlled by some rules specifying who can enter and who can see and do what in a restricted shared space. The rules are based on the position a person holds inside the organization, or/and the role he/she plays in the particular process case.

To make the above scheme work, we also need to provide a mechanism for issuing invitations to process participants to visit shared spaces and complete tasks in the frame of respective process instances/cases. There can be different solutions for this problem. The ones that we used in our practice are described in the next sections.

In the system described above, there is no information flow. A person is invited to visit a shared space and complete a task in it with the assumption that all information he/she needs is already there. In a normal business environment, a worker participates in many process instances and often in parallel. For the above scheme to work efficiently, he/she needs to understand the situation in a shared space he is visiting at a glance. This leads us to the requirement that each shared space should be highly structured, as nobody can work efficiently in unstructured shared spaces, e.g., as presented in Fig. 2. More important, shared spaces that belong to the same process type should be structured in the same way. The structure should facilitate easy understanding of in what state the process instance is and allow a person to quickly find all information related to the task at hands.

Summarizing the discussion of this section, the key to using the idea of shared spaces in BP support system is a proper structure of shared spaces.

4 Experience with a "Static" Structure of Shared Spaces

4.1 Short Description

Our initial idea for structuring shared spaces in BP support systems was to directly follow the state-oriented approach from [2]. A number of BP support systems have been implemented based on this idea, among which a system called ProBis is the most representative one [4,5]. In ProBis, a shared space consists of two parts, a static part

that reflects the structure of the underlying state space, and a dynamic part – process plan and history.

A shared space is presented to the end-user as a form/window separated in several areas by using the tab dialogues technique, see Fig. 3. Some areas of the window are standard, i.e. independent from the type of the business process, while others are specific for each process type supported by the system. Standard areas comprise such attributes and links as:

- Name and informal description of a process instance
- Links to the owner, and, possibly, the process team
- Links to the relevant documents, created inside the organization and received from the outside

The standard part of ProBis shared space includes also the task area (tab) that contains two lists, as in Fig. 3. The "to-do" list (to the left on Fig. 3) includes tasks planned for the given process instance; the "done" list (to the right on Fig.3) includes tasks completed in the frame of it. A planned task defines what and when something should be done in the frame of the process instance, as well as who should do it. In ProBis, the process plan serves as a mechanism for issuing "invitations" to attend a particular shared space. All "invitations" from all process instances are shown in an end-user's personal calendar, see Fig.4. From the calendar, the user can go to any shared space to which he was invited in order to inspect, change, or execute a task planned for him/her.

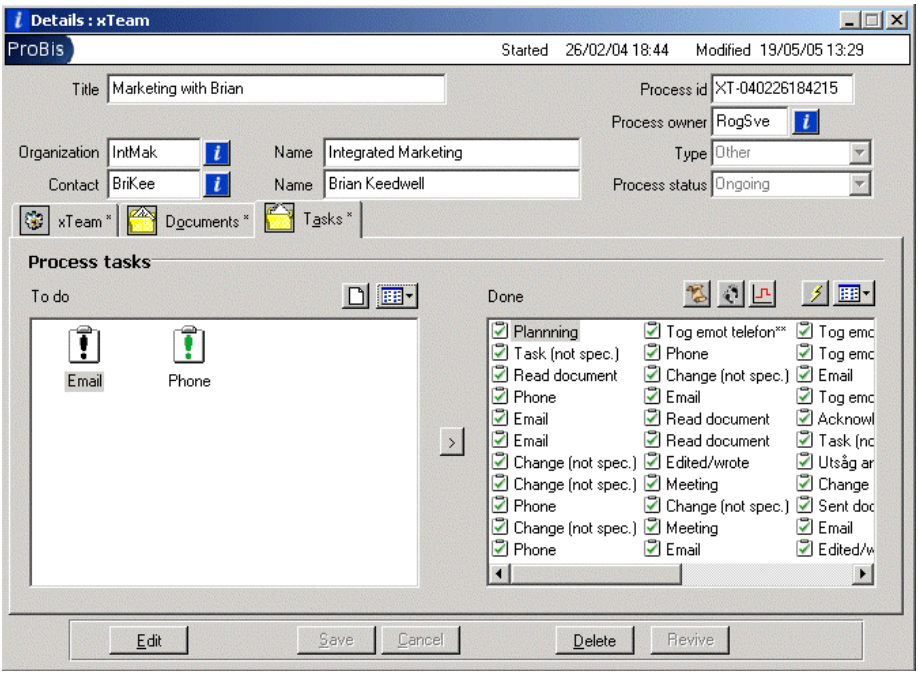


Fig. 3. A static structure of a shared space employed in BP support system ProBis (see [4])

In case of a change in the user's planned tasks, e.g., when a new task is added to some process instance and assigned to him/her, a pop-up window appears to inform the user about the change. If the user is not on-line, an email message is sent advising him/her to log in and view the changes.

The "done" list shows all events that already happened in the frame of the given process instance, independently of whether they appear there as results of planned tasks execution or as ad-hoc changes in the process state. An event (completed task) shows the date and time of when it happened, participants of the event, comments on it, etc.

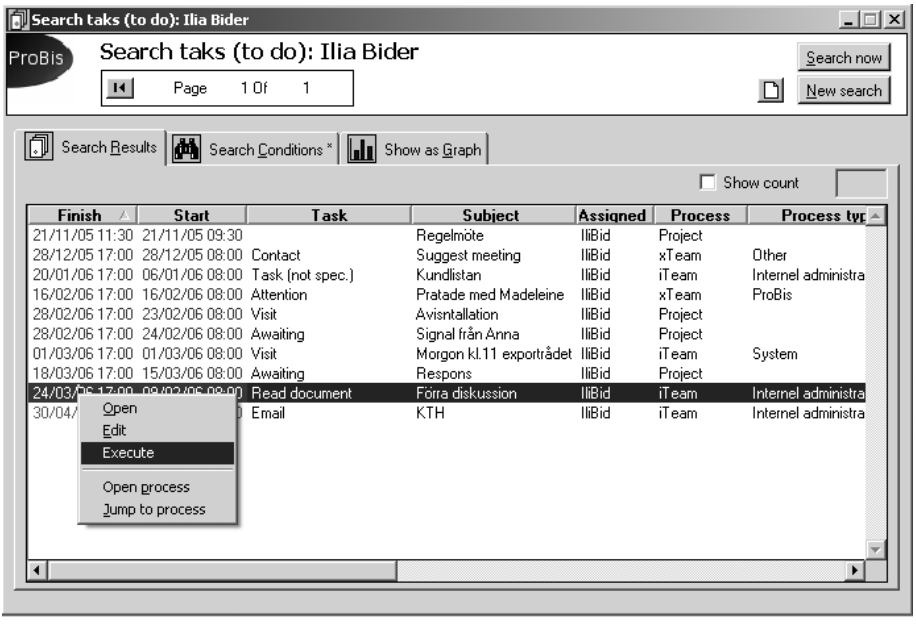


Fig. 4. A person's calendar serves as a mechanism for inviting him/her to visit shared spaces

Process participants work with the shared spaces in ProBis in the following manner. A participant comes to a shared space because a task has been planned for him/her in this space, or in the ad-hoc manner while browsing through the list of existing shared spaces (i.e. opened process instances/cases). When in the space, he/she can decide to make changes in it by changing the values of various fields, attaching new documents or persons to the shared space, etc. Any change in the shared space results in adding an event to the "done" list of the tasks tab (Fig. 3). If the change is due to the execution of some planned task, the event represents a report on the execution, otherwise the event represents some ad-hoc activity. In the simplest case, a process participant just moves his/her planned task from the "to do" list to the "done" list and presses the save button. He/she may also choose to write a report or attach a document to the event.

When changing a shared space, a participant can make changes in its plan (to-do list) by adding new tasks, or augmenting or deleting the existing ones. When inserting a new task he/she can plan it for him-/herself or to any other person. The latter serves as an invitation for this person to visit the shared space.

The above scheme provides several layers of information to a person when he visits a shared space. For example, if he visits the space to complete a planned task the following information is available to him/her:

- Task description, which includes a name of the task and its parameters. The name informs what action to complete, like contact somebody, read or write a document, etc. The parameters provide additional information, like whom to contact (link to a person), what document to read (link to a document), textual description with additional instructions, who planned the task, etc.
- Reference to the event from the "done" list that has caused this task to appear in the to-do list.
- All information already in the shared space, values of various fields, documents attached to the process instance, etc.
- "Done" list that functions as a full chronicle (log) on what has happened in the frame of the instance.
- "To do" list that provides information on what is going to happen and helps to avoid double planning.
- Full historical information about the process instance. A user visiting the shared space can see what it looked like before or after any event registered in the "done" list, or browse through the past states of the shared space one by one in the forward or backward direction.

The user visiting the shared space decides for him/herself how much information he/she needs for completing the task at hands. He/she can satisfy him-/herself with the task description, or scrutinize the whole case, including full history.

4.2 Lessons Learned

Based on our experience with ProBis, one thing is certain: a system of this kind provides a very efficient way of communication in the frame of business process instances. It is especially useful for:

- loosely structured processes, i.e. the processes for which there are no predefined ways for handling each case
- driven by a professional team that knows how to use the system quite well

There are, however, two main drawbacks with the approach when using it for more structured process or/processes that involve occasional users:

1. The dynamic aspect of business processes is poorly visualized. One needs to go through the done-list and browse throw the history to get an understanding of how a given process instance is developing in time.
2. To use the system puts some requirements on the user, as he/she needs to understand the general ideas built in the system and get some training. This means that

the system is not very friendly for newcomers and casual users. Planning as a way of issuing invitations causes the major problem here, as it is considered to be counter- intuitive. Detailed planning is not as widespread in business life as one can imagine.

We found that these two drawbacks above considerably hamper the possibility of utilization of systems like ProBis for structured processes with many occasional or untrained users. This is especially true in the current business environment where end-users more or less refuse to read manuals and demand the system being so intuitive that one can fully understand it by using try-and-error techniques. This observation has led us to rethinking the whole concept and designing a new way of structuring shared spaces that is better suited to the purposes mentioned above.

5 Experience with a "Dynamic" Structure of Shared Spaces

5.1 Short Description

Our latest system is called iPB [6]. In contrast to ProBis, it is not a ready-made process support system, but a tool (more exactly a web service) for developing such systems. In an iPB-based system, shared spaces are structured according to the process map designed for a particular process type. A process map in iPB is a drawing that consists of boxes placed in some order, see Fig. 5. Each box represents a step inside the process, the name of the step appearing inside the box (no lines or connecters between the boxes). A textual description is attached to each step that explains the work to be done in this step.

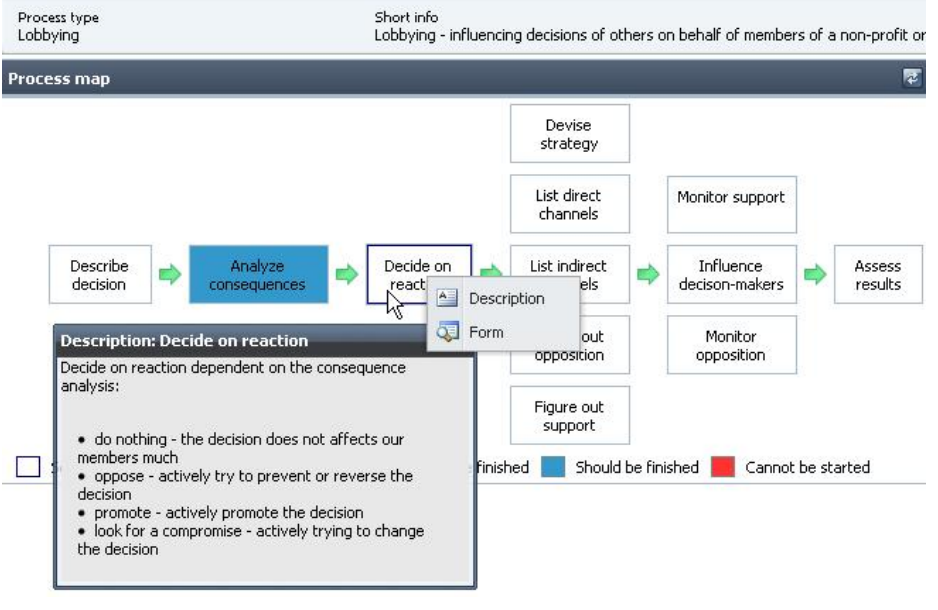


Fig. 5. A process map in iPB

Each process instance gets its own copy of the map that serves as a table of contents for its shared space. The map is used for multiple purposes: as an overview of the case, guidelines for handling the case, and a menu for navigating inside the shared space, see Fig. 6. The user navigates through the shared space by clicking on the boxes of the steps with which he/she wants to work. Not all boxes are clickable at the beginning, those that are grayed require that one or several previous steps are dealt with first, see Fig. 6.

A click on a step box redirects the end-user to a web form that assists him in completing the step, see Fig.7. The form contains text fields, option menus and radio-buttons to make choices, checkboxes, as well as more complex fields. The form may also include "static" texts that explain what should be done before one can fill some fields.

The progress in filling the step forms is reflected in the map attached to the shared space via steps coloring. A gray box means that the step form has not been filled and cannot be filled for the moment. A white box means that the step form is empty but can be filled. A step with a half-filled form gets the green color, and additional information about when the work on it has been started, and who started it. A step with a fully filled form gets the blue color, and additional information about the finish date.

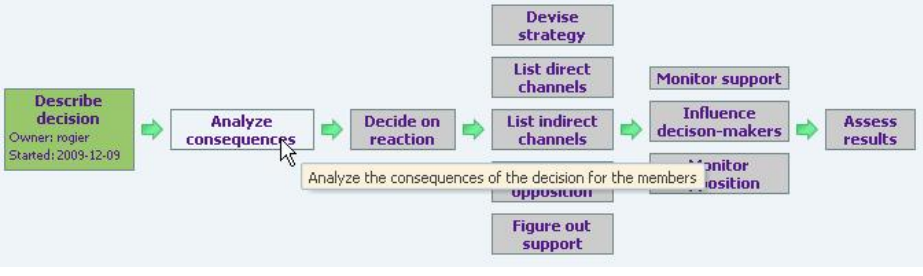


Fig. 6. The map used for structuring an instance shared space

The main way of inviting a person to visit a particular shared space in iPB is by assigning him/her to become an owner/co-owner of some step. Such an assignment results in an email message delivered to this person, and the process to appear in his/her list of "My processes". When visiting a process shared space, a person can see directly on the map what step(s) are assigned to him. Optionally, the same scheme of planning tasks as described in the previous section can be used.

5.2 Underlying Theoretical Model

From the theoretical point of view, the approach described above represents a modification of our state-oriented view on business processes [2]. The basic ideas behind this modification consist of the following:

– The total process state-space is divided into a number of subspaces called process steps. The steps are graphically represented to the end-users as boxes. Subspaces may or may not intersect. The structure of a step subspace is represented to the

end-users as a form to fill, see for example Fig. 7. Intersecting subspaces means that web forms attached to different steps may contain the same field(s). Usually, in this case, the intersecting fields can be changed only in one form; they are made read-only in the second one.

– The steps are ordered in a two-dimensional matrix that defines a recommended strategy of movement in the state space. The movement starts in the top leftmost subspace and continues in the top down left to right order. This matrix does not prohibit any other way of movement through the subspaces. For example, it allows parallel movements in several subspaces. The matrix is presented to the end-users in the form of a process map, see, Fig. 5, and 6.
– The restrictions on movement through the subspaces are defined with the help of business rules. Such a rule, for example, may require that movement in one subspace should be finished before the movement in another one can be started. Business rules are represented to the end users via gray boxes – steps that cannot be handled yet. Clicking on a gray box results in a message that explains why the box is gray, e.g. that some other box should be started first.

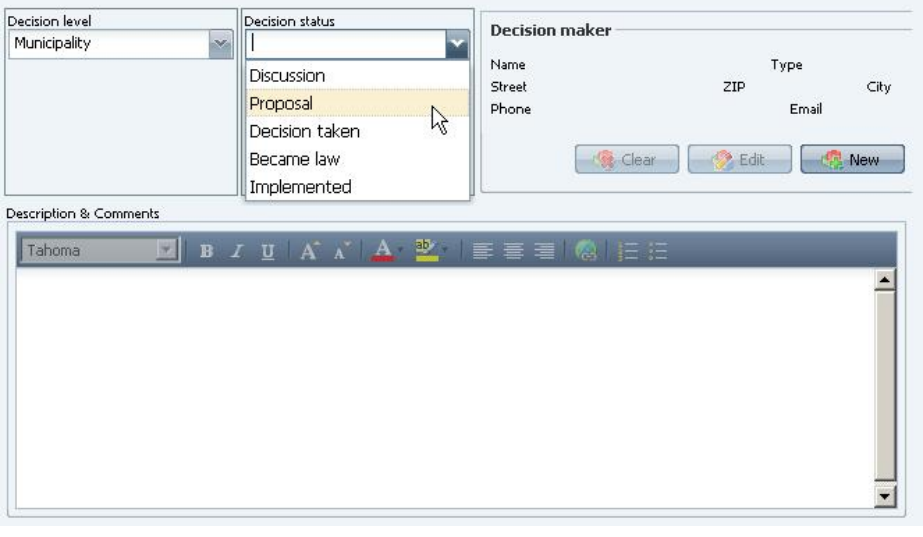


Fig. 7. A step form for the first step from Fig. 6

5.3 Lessons Learned

Our experience of introducing iPB-based systems into operational practice shows that end-users, even new ones, have no major problems in understanding the structure of shared spaces, and they learn to navigate in them very quickly. Users appreciate the idea of the multipurpose map that gives them an instant overview of the case, and simultaneously serves as a tool for navigating in the shared space. Our practical experience so far gives us a hope that we, at last, found a right approach to structuring shared space in BP support systems.

6 Conclusion

We started with the idea of introducing shared space technique from the social software into BP support systems. In the first part of this paper, we analyzed the role that shared spaces, very well known from the groupware research (see, for, example, [7]), could play in BP support using the concept of information logistics. Based on this analysis, we concluded that a shared space could serve as a kind of a "construction site" that contains all information related to a given process instance/case. For the idea to work in practice, the shared spaces need to be properly and uniformly structured. This structure should reflect the peculiarities of the given process type(s) that the system is aimed to support.

In the second part of this paper, we described two cases of realization of the idea in real BP support systems. The first case represents a system with rich functionality, which, however, lacks proper visualization for structured processes, and requires extensive training before it can be used in operational practice. The second system is highly visual and easy to learn, though it, for the moment, lacks some functionality that can be found in the first system, e.g. full history.

Both cases are based on the state-oriented view on BP from [2]. In the first case, the original idea has been used as is. The second case exploited a modification of the original idea that consists of splitting the total process state space into subspaces, and introducing some order between them. The order is introduced in two ways, via relative positioning (recommended order), and via business rules (hard restrictions). The modification makes it possible to represent the process space as a map that is somewhat similar to the traditional workflow, though it has different semantics. This map makes it possible to achieve much better visualization of shared spaces of structured processes to the end-user than when using the original approach.

Our experience of introducing BP support systems into operational practice shows that the end-users appreciate the support provided by the systems built on the modified state-oriented view. Thus, we conclude that the idea is promising but needs to be developed further. Our future plans include developing the split state space model in more details theoretically as well as continuing to further develop software based on it.

The theoretical part concerns relationships between subspaces, which will enrich iPB with a more elaborated system of business rules. The practical part consists of moving the rich functionality built in ProBis to iPB to satisfy the future demands from the more experienced users.

Acknowledgements. This paper would have never been written without considerable efforts of the team of developers who have designed and implemented both ProBis, and iPB. We are especially thankful to Tomas Andersson, Alexander Durnovo, Alexey Striy and Rogier Svensson.

References

[1] Nurcan, S., Schmidt, R.: Introduction to the First International Workshop on Business Process Management and Social Software (BPMS2 2008). Business Process Management Workshops 2008, pp. 647–648. Springer, Heidelberg (2009)

[2] Khomyakov, M., Bider, I.: Achieving Workflow Flexibility through Taming the Chaos. In: 6th international conference on object oriented information systems, OOIS 2000, pp. 85–92. Springer, Heidelberg (2000)
[3] Bider, I.: New Logistics for Administrative/Business Processes. White paper, IbisSoft, 14 p. (2006), http://www.ibissoft.se/whitepapers/newlogistics.pdf
[4] Andersson, T., Bider, I., Svensson, R.: Alignig people to business processes. Experience report. In: Software Process: Improvement and Practice (SPIP), vol. 10(4), pp. 403–413. Wiley, Chichester (2005)
[5] Bider, I., Striy, A.: Controlling business process instance flexibility via rules of planning. International Journal of Business Process Integration and Management (IJBPIM), Inderscience 3(1), 15–25 (2008)
[6] iPB Reference Manual 8on-line documentation. IbisSoft (2009), http://docs.ibissoft.se/node/3
[7] Takemura, H., Kishino, F.: Cooperative work environment using virtual workspace. In: Proceedings of the 1992 ACM conference on Computer-supported cooperative work, pp. 226–232. ACM, New York (1992)

Mirror, Mirror on the Wall, Can I Count on You at All? Exploring Data Inaccuracy in Business Processes

Pnina Soffer

University of Haifa, Carmel Mountain 31905, Haifa, Israel
spnina@is.haifa.ac.il

Abstract. Information systems in general and process aware information systems in particular support the execution of business processes. This support is based on the assumption that the information system truly reflects the state of the domain where the process takes place. Based on this assumption, humans do not need to directly "sense" the state of affairs. Rather, decisions are made based on the state as reflected in the information system. This paper explores the situation where this assumption does not hold, namely, the situation of data inaccuracy. In particular, it formalizes data inaccuracy and addresses three questions: (a) how is data inaccuracy manifested in a process? (b) what are the expected results of data inaccuracy? (c) how can robustness to data inaccuracy be increased? The understanding gained with respect to these questions should form a basis for designing processes to be more robust, avoiding problems due to data inaccuracy.

Keywords: Data inaccuracy, Process design, Generic Process Model.

1 Introduction

Business process management has attracted the attention of both business and academia in the past two decades. Typically, the focus of research has been on process aware information systems and workflow management systems. However, business processes are usually performed by humans who use resources. Information systems provide different levels of support to business processes. The basic support is a simple reflection of the activities that are performed and their effect on the state of the organization and its environment. At the high end of support, process aware information systems enact, coordinate, and manage these activities, while reflecting their effect. At all levels, the basic assumption underlying business process support, is that the information system truly reflects the state of affairs. Based on this assumption, humans do not need to physically sense the current state for deciding what activity to perform at a given moment and how to perform it. However, it is well known that the information which exists in an information system is not always completely reliable [1].

The question is what would happen to the business process if and when the information its execution relies upon does not truly reflect reality; will it be able to complete? What would be the results? Clearly, there is no one answer to this question, as the answer is highly situation dependent. To illustrate, consider the following two

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 14–25, 2010.

scenarios. The first one addresses a process of quotation preparation, which includes the analysis of work and materials required for fulfilling the customer's needs, estimating their cost, and determining the price to be charged based on business considerations. The goal of the process is reaching a state where the quotation is ready and sent to the customer. Now assume the material cost has been falsely recorded in the information system and does not reflect the actual price. This would not stop the process from achieving its goal, and the result of this inaccuracy may even remain unknown. It might, however, have an effect on the business value achieved by the process (under-pricing will harm profitability, while over-pricing might lead to rejection by the customer).

The second scenario addresses a process where a package is sent by a courier to an address given by a customer. The goal of the process is reaching a state where the package is received at the delivery destination and confirmed by the recipient. Now assume the destination address recorded is incorrect (e.g., Birmingham Alabama instead of Birmingham UK). Clearly, it would be impossible for the process to achieve its goal.

While data-aware process design has been investigated to some extent [7][10][9][15][16], the focus of attention has been on combining data flow with activity flow, and avoiding design time errors. A data centric perspective at run time has also emerged [4][5][6][17], allowing changes to the process at run time while maintaining its soundness. However, systems of this kind, like "traditional" process-aware information systems, depend on the quality of the data and build on the assumption that the data is reliable. To the best of our knowledge, avoiding runtime problems that may arise due to data deficiencies in business processes has not been addressed so far. Hence, before solutions can be developed, some understanding of the considered phenomenon should be achieved.

This paper aims at exploring the situation where the information system does not truly reflect the state of a domain where a process takes place, namely, the situation of data inaccuracy [18]. In particular, we ask the following questions: (a) how is data inaccuracy manifested in a process? (b) what are the expected results of data inaccuracy? (c) how can robustness to data inaccuracy be increased? We address these questions using the conceptual framework of the Generic Process Model (GPM) [11][12], which is anchored in an ontology that depicts domain behavior.

The remainder of the paper first provides a formalization of data inaccuracy and some relevant concepts. Then the three questions presented above are addressed. After discussing the questions based on the GPM conceptual framework, we demonstrate the suggested ideas using an example. Finally, conclusions and future work are presented.

2 Formalizing Data Inaccuracy Concepts

This section formalizes the notion of data inaccuracy, based on the Generic Process Model (GPM) [11][12]. GPM is ontologically-based in that it uses a specific set of fundamental constructs developed originally by Bunge [2][3] for modeling "real world" domains. The basic set of concepts includes – things, properties, composition, attributes, states, events, and laws. A process is considered a "trajectory" of states in a

certain domain comprising things. An important aspect is the notion of goal which is a set of states the domain is intended to reach at the end of the process. The left-hand side of Fig. 1 depicts the main concepts used for representing the dynamics of a process in the real world.

According to GPM, a (real world) process takes place in a *domain*, which is a composite *thing*, including things (e.g., courier, customer) and their interactions. The *state* of a thing (and of the domain) is the set of values of its *state variables* (properties – e.g., address). States are changed by *events* (e.g., package sent), which can be *external* to the domain or *internal*, in which case they are governed by the *law* of the domain. The real world is reflected in the information system, depicted in the right-hand side of Fig. 1, where dotted lines denote a representation relation. Things are represented by *information objects*[1], which have a defined set of attributes or *data items*, whose value reflects the value of a corresponding state variable at a moment in time. Hence, the values of all data items at a moment in time reflect the current state of the domain. Events are reflected by *functions* of the information system (can be some computation or input made by a user), which modify values of data items. Note, these functions can include the creation or deletion of an information object with all its set of data items. Finally, in systems that include an executable *process model*, it is a reflection of the domain law, according to which the system functions are executed.

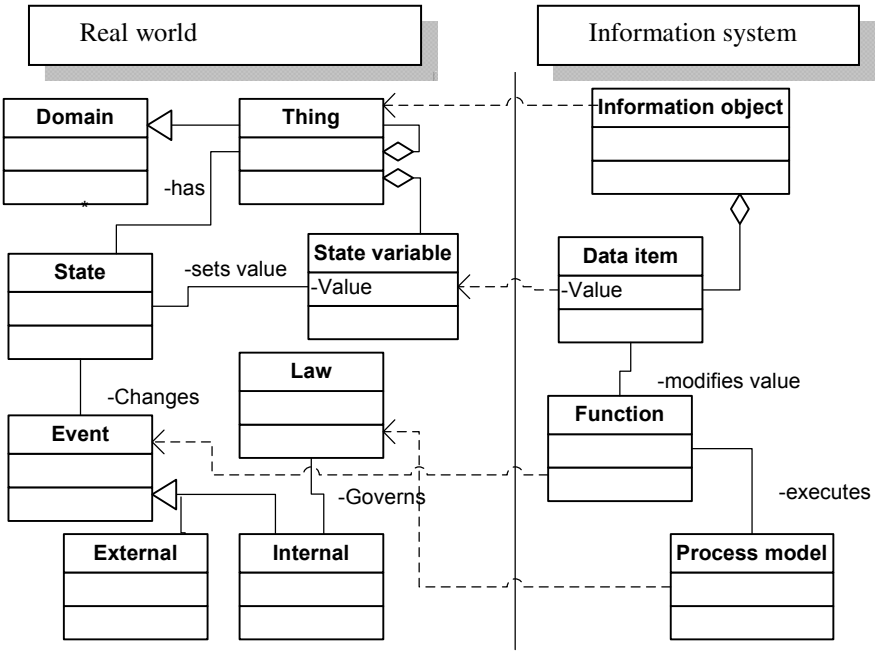


Fig. 1. Real world process and its reflection in an information system

[1] Note, there are also information objects that reflect the history of things (e.g., transaction).

The correspondence between a domain and its representation in an information system is formalized in the following definition.

Definition 1. Let $X=\{x_1, x_2,\ldots x_n\}$ be the set of state variables of the domain and $DI=\{d_1, d_2,\ldots d_n\}$ be the set of data items in the information system. $R:X\rightarrow DI$ is a function such that $d_i=R(x_i)$ implies that d_i is the data item that reflects the state variable x_i. We denote the couple $<x_i,d_i>$ where $d_i=R(x_i)$ a *corresponding* couple.

Note that, according to our notation, while x_i stands for a state variable, x_i is its value; similarly, d_i is the value of the data item d_i. As an example, consider the inventory level of an item x_i whose real value at a moment in time is x_i. It is reflected by a corresponding data item d_i in an information system, whose value at that moment is d_i. Changes in the inventory level are reflected as updates in the value of the corresponding data item.

We are interested in exploring situations where the domain state is not truly reflected by the information system state. We use the correspondence relation to define these situations.

Definition 2. A domain state $s=(x_1, x_2,\ldots x_n)$, is truly reflected by an information system state $s_R=(d_1, d_2,\ldots d_n)$ iff $x_i=d_i$ $\forall$ corresponding couple $<x_i,d_i>$. *Inaccuracy of data* is a situation where $\exists$ a corresponding couple $<x_i,d_i>$ such that $x_i\neq d_i$.

Considering the inventory example above, when $x_i\neq d_i$ the information system does not accurately reflect the real inventory level.

Definition 2 is one basis for the following discussion of data inaccuracy and its effects. The second basis is the GPM notion of independent sub-domains which we briefly introduce here (more details can be found in [13][14]).

Definition 3. A *sub-domain* is a part of the domain described by a subset of X.

When looking at a sub-domain at a moment in time, each state variable of the sub-domain has a value, and together they define the state of the sub-domain. This state is actually a *projection* of the state of the entire domain on the sub-domain. The projection of a domain state s over sub-domain Z is denoted as $s_{/Z}$. When the domain transforms and changes its state, a sub-domain can change accordingly. We then say that the domain law is projected on the sub-domain. However, for a given sub-domain, the changes in its state variable values may depend on state variables outside the sub-domain. For example, consider a sales clerk and a sale order as a sub-domain in an order fulfillment process. Accepting or rejecting an order depends on state variables outside this sub-domain, such as inventory levels, production capacity, and the customer's credit. Observing this sub-domain in isolation, the law might seem to be unpredictable.

In some cases, the projection of the law over a sub-domain is such that the transformations depend only on state variables within the sub-domain itself. Consider, for example, a sub-domain of a warehouse receiving goods. The goods may arrive from a supplier or from a sub-contractor, but the projection over the warehouse would not be affected by state variables outside the warehouse itself. In other words, in this case the law is a function mapping states of the sub-domain to states of the sub-domain.

A given unstable state of the sub-domain will always map in the same way, independent of the state of the whole domain, and hence independent of the states of other sub-domains. We will then say that the sub-domain behaves *independently*. Partitioning of the domain into independently-behaving sub-domains is often a consequence of different actors acting in the domain. These actors can be people, departments, machines, computers and combinations of those.

Definition 4. A sub-domain D^1 of D will be called an *independently behaving* (in short an *independent*) sub-domain iff the law projection on D^1 is a function.

Note that a sub-domain can be independent at some sets of states and not at others. Independent behavior of sub-domains is discussed in [13][14] as a necessary condition for concurrency. When sub-domains become independent, they may start operating concurrently; this is represented in process models as a split point. When they reach a state where their (projected) law is not independent, this would be represented as a merge point in a process model. We term the sub-domain which will continue transforming at the merge the *continuation* sub-domain. Various kinds of merge behaviors exist, differing from each other in the conditions that activate the continuation of the process [13]. One known behavior, that can merge sub-domains that operate concurrently, is when the process continuation is activated only when all the independent sub-domains have reached the merge (ceased being independent). This is called a *synchronizing* merge, as formalized in the following definitions.

Definition 5. A set of states S_{sp} is a parallel split iff there exist at least two sub-domains such that for every state in S_{sp} each sub-domain is independent and is in an unstable state.

Definition 6. Let $D^k \subset D$, k=1...n be independent sub-domains operating concurrently following a split point S_{sp}. A set of states S_{me} is a *synchronizing merge* iff a continuation sub-domain D^C exists, such that:

(1) Each sub-domain D^k has at least one unstable state projected to from a domain state for which the continuation sub-domain D^C is stable, and at least one stable state u_k projected to from a domain state where the continuation sub-domain D^C is unstable.
(2) For each sub-domain, there is at least one u_k that projects onto the same unstable state of D^C as all other sub-domains.
(3) There are no other unstable states of the D^C projected into by a state in the merge set S_{me}.

The stability in (1) assures each sub-domain will "wait" for the others before continuation is activated. (2) and (3) assure that D^C will only begin changing when all sub-domains have "arrived" at their "appropriate" states (u_k). Together, these conditions assure synchronization.

To demonstrate the definition, consider a process where several parts need to be manufactured for a product to be assembled. When a production order is given, the domain enters a split state where each part is made by a separate production cell. When each cell has completed making the part, the cell "rests". Only when all cells

completed (hence each is at rest – in a stable state), the domain enters a state where the product can be assembled, namely, the continuation sub-domain is activated.

Synchronization of independent sub-domains is a key concept in our discussion of the consequences of data inaccuracy in the following section.

3 How Is Data Inaccuracy Manifested in a Process?

For discussing the potential consequences of data inaccuracy, let us return to the two scenarios introduced in Section 1. In the first scenario an inaccurate value of material cost was recorded in a quotation preparation process. The process continued and completed without detecting the inaccuracy. In the second scenario, an inaccurate delivery address was recorded in a package delivery process by a courier, preventing the process from achieving its goal and completing.

These two scenarios are clearly different with respect to the effect of inaccuracy. However, the fundamental underlying difference that causes different results of inaccuracy needs to be explored. In particular, we wish to explore the circumstances under which inaccuracy would be acknowledged in a process.

To address this question, we now abandon the separated view introduced earlier of the "real world" domain and its reflection in the information system. Rather, we look at the process domain as including both the real world and the information system. Formally, our process domain will be represented by the set of state variables $X^D = X \cup DI = \{x_1, x_2, \ldots x_n, d_1, d_2, \ldots d_n\}$. Addressing this extended domain, we rely on the notion of independent dub-domains to explain the consequences of data inaccuracy.

In the second scenario above of a package delivered by a courier, the extended domain includes "real" state variables ($x_1, x_2, \ldots x_n$) such as the ordering customer, the delivery destination, etc., and state variables holding their reflections ($d_1, d_2, \ldots d_n$). Once the order details are recorded, the process progresses at a sub-domain which includes some "real" state variables (e.g., means of transportation) as well as reflection data items (e.g., the delivery address). This sub-domain acts independently of the sub-domain holding the real delivery destination. Assuming the delivery address has been incorrectly recorded, the independently acting sub-domain would not be affected by the real delivery address. Relying on the corresponding data item, transportation is arranged, and the package is loaded and sent. At the same time the other (recipient) sub-domain may transform, preparing for the package to arrive. The two sub-domains should cease being independent and synchronize at the merge point, when the package arrives. At the merge point, the domain state is expected to be as follows: the courier sub-domain has brought the package to its destination and became stable; the recipient sub-domain is stable waiting for the package; the process continuation would be triggered when the recipient gets the package. However, due to the mismatch between the real destination and its reflection, the actual state reached is not a state projected to from the merge, since the package is not at the same place as the recipient. Hence, synchronization cannot take place and the process cannot continue.

In contrast, consider the first scenario above where material cost was falsely reflected in a process of quotation preparation. While relying on the reflected value of material cost, the process does not include any step where "synchronization" with a

sub-domain including the actual value is required. Hence, the process may achieve its goal without recognition of the error that has been made.

Summarizing this discussion, data inaccuracy becomes apparent when the sub-domain including the reflected value of a state variable is synchronized with the sub-domain which includes the real value. Such synchronization can be a result of the required course of the process, typically when physical operations are performed.

4 What Are the Expected Results of Data Inaccuracy?

In the courier example discussed above, data inaccuracy resulted in a situation where the process got stuck and could not achieve its goal. However, this is not necessarily always the case. Different situations might lead to different results, and it is important to be able to characterize and distinguish the cases where the results could be severe.

In the following discussion, summarized in Fig. 2, we assume that the process includes a synchronization point, so inaccuracy can potentially be acknowledged.

One factor that may affect the results of inaccuracy is the existence of multiple paths in the process. There are three possibilities for the law to address a state variable (and its corresponding data item). First, the law may "use" its value on every possible path leading to the process goal. In this case, the value is *mandatory* for the process to complete, and, based on our assumption, synchronization with the real value will be required before or at the process goal.

Second, the law may use its value on a subset of the paths leading to the goal. In this case the value is *optional* for the process completion. If the process takes a path where the value is not used at all, inaccuracy will not affect the process. Otherwise, if the process takes a path where the value is used, once inaccuracy is recognized, it may be possible to roll the process back [8] to a point where a different path can be taken. This might have a negative effect on the business results of the process. For example, if the recipient's cell phone number is discovered to be wrong when delivery needs to be coordinated, some other means of communications may be used (e.g., email). This might take more time than if the number was correct.

Third, once recorded, the value of a state variable might not be used by any transition in the process, and is hence *redundant*. In this case, the value might be needed for other processes in the organization. We may thus assume that analyzing the consequences of its inaccuracy should be done with respect to those processes and not in the analysis scope of the process under consideration.

Even when considering a mandatory variable or one that is used on the chosen process path at runtime, two possibilities exist as to the results of synchronization with the real value. (a) Due to inaccuracy, the process would not reach a state where synchronization is possible. This is the above described case of the courier process. (b) Despite the inaccuracy, the process reaches a merge state, namely, synchronization is possible, and the process can progress towards achieving its goal.

In the latter case, two other possibilities exist. First, it might be that the granularity at which the law operates in indifferent to the inaccuracy in the represented value. For example, consider an inaccurate value representing inventory level, which should synchronize with the real inventory level when material needs to be issued. The law

relates to the amount that needs to be issued. If the real inventory level is above the required amount, then the process can continue and the inaccuracy may not even be acknowledged.

Second, it might be that facing the real value, the process would take a path which is different than the one planned and still achieve its goal. In this case the consequences might be different in terms of business performance measures. In the courier process described above, assume delivery was planned based on a wrong transportation schedule. Once the error is identified, special delivery can still be made, but at higher costs.

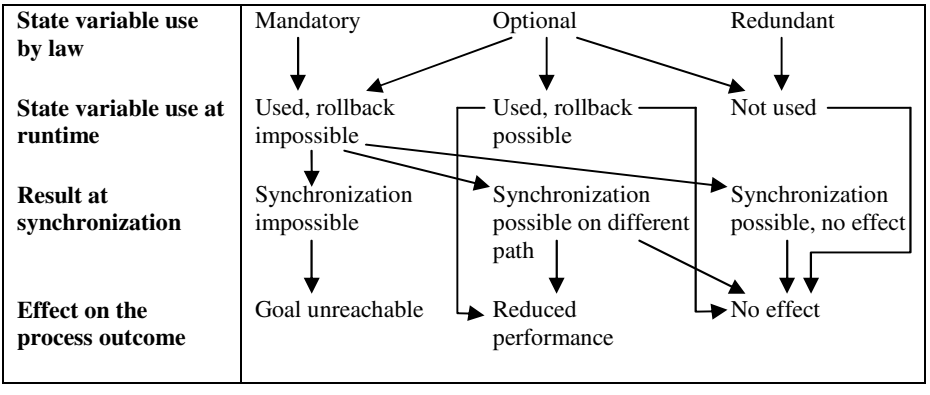


Fig. 2. Possible results of data inaccuracy

5 How Can Robustness of Processes to Data Inaccuracy Be Increased?

Data inaccuracy is clearly related to runtime of processes. However, it is possible to take this possibility to account at design time, and design a process to be more robust to errors that may occur in the represented values.

As discussed above, inaccuracy is not necessarily discovered at the moment it occurs. It might be discovered only later on in the process, after many actions have been performed based on an incorrect value, sometimes when it is too late to restore the process to a state from which its goal can be achieved. In other cases, inaccuracy might not even be recognized after the process has reached its goal (e.g., the quotation preparation process), but business results would be harmed.

The following issues come up from our above discussion: (a) Inaccuracy can be discovered when the sub-domain including the reflected value of a state variable is synchronized with the sub-domain which includes the real value. (b) The result of inaccuracy is expected to be different for different state variables that participate in the process.

Considering point (a), such synchronization can be a result of the required course of the process, typically when physical operations are performed. However, it can also be added to the process as a step intended to verify the reflected values, in order to

prevent execution based on false information. In the quotation preparation example, such verification could be achieved by collecting information from different sources (e.g., supplier and inventory records) and comparing the values obtained. It is even not necessary for the verification to use the real value. Rather, verification might be achieved through comparison with similar quotations, or through a review by independent experts. In general, verification would be a step where the process cannot continue unless the values that relate to the IS reflection match some values that are *independent* of that reflection.

Still, we do not propose to introduce verification steps for each state variable value in the process. This would result in inefficiency. Rather, based on issue (b) above, we should identify the state variables whose potential inaccuracy results would be most severe, and make sure their value is verified before any harm is done.

6 Demonstration

This section demonstrates the above presented ideas by applying them to an example process. To visualize the process, we adapt the Workflow nets with Data (WFD-nets) notation proposed by [10][16] for data incorporated workflow models. A WFD-net is a Workflow net, namely, a Petri net with one initial place and one final place, whose transitions are annotated to represent data operations. The notation relates to three operations: write (wt), read (rd), and delete (del). To support our purposes, we add a fourth one, Synchronize (sn), denoting that the real value of a state variable is synchronizing with the process at that transition. Note that WFD-nets also include Guards, which are guarding functions that specify the conditions for selecting a specific path. In our example these are not specified.

Our example process is of organizing outdoor social activities, typically ordered by companies for groups of their employees. When a customer's order is received, the details are agreed upon and recorded. These include the planned date, the location, the type of activity (it can be some sporting activity, a designated workshop, or a guided tour), specific requirements regarding the food (style, number of participants), and required means of transportation. Transportation arrangements are not always needed, and it is possible that the participants will arrive at the meeting location by themselves. On the planned date, physical preparations in the location are made, and the participants arrive (by themselves or by the coordinated transportation). After the social activity is executed, payment is made. The process model is depicted in Fig. 3.

To demonstrate our ideas, we shall consider possible errors in the representation of some of the state variables in the process (each one by itself, not in combination with others).

Planned date – assume the planned date is falsely recorded. The synchronization with the real value of this state variable is on the planned date itself, either when transportation waits for the participants or when the activity itself should take place. Clearly, if the activity is organized for a date which is different than the one the participants prepared for, synchronization cannot take place and the process gets "stuck", unable to achieve its goal.

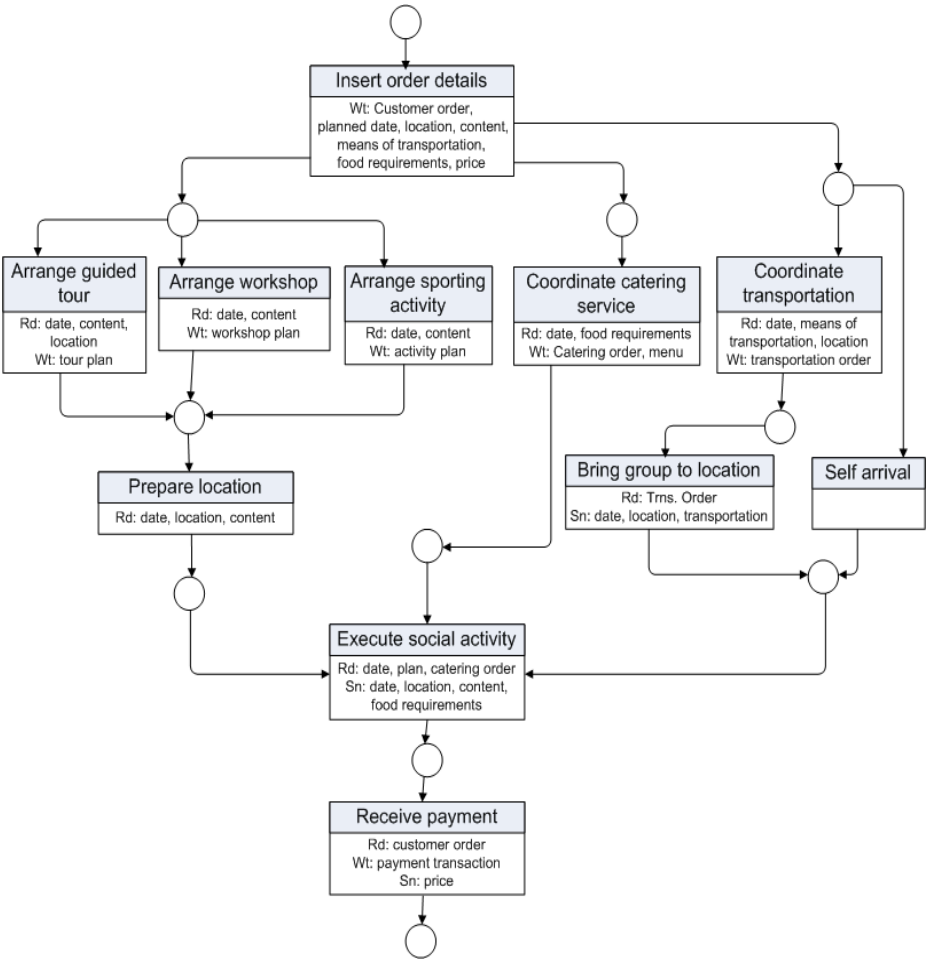


Fig. 3. The social activity organization process

Means of transportation – assume the customer required some means of transportation, while the record in the information system is for self arrival. Synchronization is when the participants would wait for transportation to arrive, which would not take place. However, this path is optional in the process model. Once the problem becomes apparent, it may still be possible to "roll back", so the participants use their own vehicles and arrive at the location. This would take time and reduce their satisfaction, so the process would be able to achieve its goal, but with lower performance indicators.

Food requirements – assume a requirement for vegetarian food was falsely entered. Synchronization of this state variable is while the social activity is being executed, so it would probably be too late to roll back and prepare other kinds of food. Again, this would not stop the process, but reduce the level of participants' satisfaction.

Price – assume there has been an error in the recorded price. The synchronization of this state variable is when payment is made. Payment would be possible (so the process can still achieve its goal), but profitability might be harmed.

The analysis of inaccuracy with respect to these state variables demonstrates the different possible consequences. Clearly, the most severe would be inaccuracy of the date. Still, inaccuracy would have harmful results for all the other state variables as well. One of the reasons for this is that synchronization only takes place when, in most cases, it is too late to recover without any loss. A possible solution would be to introduce a synchronizing step earlier in the process. For example, it would be possible to send the plans to the customer for approval once they are ready. This would constitute an early synchronization point, and enable detecting inaccuracy when it is still possible to change the plans and avoid any damage to the process.

7 Conclusions

A lot of effort has been devoted to the design of sound process models and the development of process aware information systems to support them. However, these all depend on the quality of the representation in the information system. Humans have learnt to rely on information systems and to make decisions based on the information they provide about the state of the world. Hence, designing business processes to be robust and resilient to deficiencies and inaccuracy in the data stored in the information system is an important challenge.

This paper is a first step towards systematically addressing the possibility of run-time data deficiencies when designing processes. As a first step, it conceptualizes the problem and provides some understanding of its underlying mechanism. It also motivates further investigation of this issue by highlighting the consequences and possible results of data inaccuracy.

As future work, a lot has yet to be done and many questions are still unanswered. In particular this would include identifying the "weak links" – the state variables whose verification is crucial for the process to achieve its goal. Following this, the challenge remains to provide methods that would support the design of processes to be robust and avoid problems related to data inaccuracy.

Acknowledgement. Some of the ideas presented in this paper are the result of discussions with Michael Vaknin, who brought the issue of data inaccuracy to my attention.

References

[1] Agmon, N., Ahituv, N.: Assessing data reliability in an information system. Journal of Management Information Systems 4(2), 34–41 (1987)

[2] Bunge, M.: Treatise on Basic Philosophy: Ontology I: The Furniture of the World, vol. 3. Reidel, Boston (1977)

[3] Bunge, M.: Treatise on Basic Philosophy: Ontology II: A World of Systems, Boston, vol. 4. Reidel (1979)

[4] Cohn, D., Hull, R.: Business Artifacts: A Data-centric Approach to Modeling Business Operations and Processes. IEEE Data Eng. Bull. 32(3), 3–9 (2009)

[5] Künzle, V., Reichert, M.: Towards Object-aware Process Management Systems: Issues, Challenges, Benefits. In: Proc. BPMDS 2009, pp. 197–210 (2009)
[6] Müller, D., Reichert, M., Herbst, J.: A New Paradigm for the Enactment and Dynamic Adaptation of Data-Driven Process Structures. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 48–63. Springer, Heidelberg (2008)
[7] Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow Data Patterns:Identification, Representation and Tool Support. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, pp. 353–368. Springer, Heidelberg (2005)
[8] Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Workflow exception patterns. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 288–302. Springer, Heidelberg (2006)
[9] Sadiq, S.W., Orlowska, M.E., Sadiq, W., Foulger, C.: Data Flow and Validation in Workflow Modelling. In: Fifteenth Australasian Database Conference (ADC), Dunedin, New Zealand. CRPIT, vol. 27, pp. 207–214 (2004)
[10] Sidorova, N., Stahl, C., Trcka, N.: Workflow Soundness Revisited: Checking Correctness in the Presence of Data While Staying Conceptual. In: Proceedings of CAiSE (to appear, 2010)
[11] Soffer, P., Wand, Y.: Goal-driven Analysis of Process Model Validity. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 521–535. Springer, Heidelberg (2004)
[12] Soffer, P., Wand, Y.: Goal-driven multi-process analysis. Journal of the Association of Information Systems 8(3), 175–203 (2007)
[13] Soffer, P., Wand, Y., Kaner, M.: Semantic analysis of flow patterns in business process modeling. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 400–407. Springer, Heidelberg (2007)
[14] Soffer, P., Kaner, M., Wand, Y.: Assigning Ontology-Based Semantics to Process Models: the Case of Petri Nets. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 16–31. Springer, Heidelberg (2008)
[15] Sun, S.X., Zhao, J.L., Nunamaker, J.F., Liu Sheng, O.R.: Formulating the Data Flow Perspective for Business Process Management. Information Systems Research 17(4), 374–391 (2006)
[16] Trcka., N., van der Aalst, W.M.P., Sidorova, N.: Data-Flow Anti-Patterns: Discovering Dataflow Errors in Workflows. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 425–439. Springer, Heidelberg (2009)
[17] Vanderfeesten, I., Reijers, H.A., van der Aalst, W.M.P.: Product-Based Workflow Support: Dynamic Workflow Execution. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 571–574. Springer, Heidelberg (2008)
[18] Wand, Y., Wang, R.Y.: Anchoring data quality dimensions in ontological foundations. Communications of the ACM 39(11), 86–95 (1996)

Ontology Driven Government Services Inventory and Business Process Management

V. Necati Sönmez, Mustafa Canlı, Selim Gökçe, Mikail Ünver, and A. Nusret Güçlü

Stratek, ODTÜ Teknokent, Gümüşbloklar B7, Ankara, Turkey
Turksat A.S., 40 KM, Konya Yolu Gölbaşı Ankara, Turkey
{vedat,selim,mikail,nusret}@tns.com.tr

Abstract. In this paper, the experience obtained in the e-government project of Turkey is described, which consisted of three parts: inventorying all services and processes in all governmental agencies; modeling several of the processes identified with the help of Extended Event Driven Process Chain technique; and execution via transforming the model to a Business Process Management tool that allowed business activity monitoring. Due to size of the project, and the limitation of the paper, main focus will be on the inventorying. The paper first discusses the ontology design for Public Services in Turkey and the process of developing the ontology based services inventory management, as the basis for e-Government gateway. Then the discussion follows on how this ontology can establish the basis for the service-based process design and finally the pilot implementation for the selected citizen-centric processes is presented.

Keywords: public services ontology, government services inventory, service oriented process management.

1 Introduction

Turksat is the public enterprise charged with the management of e-Government gateway, namely the turkiye.gov.tr site in Turkey. It is essential that the electronic services be provided on top of common understanding of the public value delivery concepts, and as per the Information Society Action Plan [*1*], the Administration Development Agency (IGB) is responsible for the management of the inventory of government services, provided to the citizens, businesses, other public agencies and to the employees of the agency. IGB first designed an Excel spreadsheet to record the information on existing services and sent the template out to all agencies to be filled in, which resulted in thousands of lines due to the lack of common understanding as to what a service is. Following a two year study, the resulting consolidated document was not consistent, contained different levels of details, lacking direct service delivery tasks, and the need for a controlled vocabulary was immediately recognized. Then, IGB subcontracted Turksat to deliver the inventory, based on international standards.

A team of experts, led by team leader delivered the Public Services Inventory through an ontology based model, capturing details in a comprehensive field study in

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 26–38, 2010.

Erzincan. This work resulted in only 950 services, with the exception of port management, and customs. After establishing the model for the Public Services Ontology, the team also modeled three important services to the citizens using extended Event Driven Process Chain model [*2*], applied the process models to a Business Process Management engine, and delivered the running portal structure with almost no direct coding for the software development.

In this paper, the authors would like to share their experience and the models developed within this 6-month study, which will establish the basis for all the e-Government services in Turkey. To complete the inventorying part, the analysis team created a high-level view on the agencies functioning that ties up such concepts as Service, Business Process, Laws and Regulations, etc.

1.1 Ontology and the E-Government

Ontology is the science of analyzing structures of objects, properties, events, process and relations in every area of reality [*3*]. Ontology in the knowledge domain means specification of conceptualization. That is, ontology is a description of the concepts and relationships that can exist for an agent or a community of agents.

Ontology models are designed to be used as a medium of knowledge sharing and reusing; together with a set of individual instances of classes (concepts), it constitutes a knowledge base. Common components of ontology models include:

- Classes: collections, concepts, kinds of things that have certain things in common
- Individuals: instances, objects, or elements of classes.
- Attributes: aspects, properties, features, characteristics, or parameters that objects (and classes) can have
- Relations: Properties, slots, ways in which classes and individuals can be related to one another

The ontology models were developed:

- to share common understanding of the structure of information among people or software agents,
- to enable reuse of domain knowledge,
- to make domain assumptions explicit
- to separate domain knowledge from the operational knowledge, and
- to analyze domain knowledge [*4*].

Within the context of the aforementioned descriptions, ontology model is a must for any domain of knowledge that has a wide range of contributors and agents. That's why a concrete ontological model of government services will be a basic starting point for:

- developing a common explicit model for all government services,
- sharing the same types of methodology among all agents of public services,

- business analysis and redesign of government services in order to eliminate redundant tasks and reduce bureaucracy, which is a common problem in public services worldwide,
- establishing a concrete infrastructure for e-government applications, and
- being able to perform an impact analysis among the objects of the model, to the extent that, the administration should be able to answer questions such as what happens if an article of a certain regulation is changed, what would be the effect of this change on the processes, authorities and other classes of the public services; and what needs to be done at the organizational, data, legislative, and process levels to be able to move the service delivery channel to electronic or mobile media.

1.2 E-Government, One-Stop Public Service Delivery and Erzincan Province Pilot Project

The efforts towards establishing a one-stop service point for all public services has a long history in Turkey, in fact over ten years, and some important progress has already been made, but it's far from being adequate. One of the shortcomings of these efforts is that, there has not been any attempt towards modeling all government services which will include all regulations, responsibilities, processes, and other objects within the public value delivery chain, with their relations and attributes for every government service. Such type of a model will form a structured public service inventory and will form the knowledge base among all stakeholders in the management of public services as well as the process and data integration within e-government.

Erzincan Province Pilot Project (EPP) was planned by IGB through Turksat to serve the ultimate need for the development of such a model in the area of public management and to achieve e-government integration and management.

The EPP Project was implemented in cooperation with Turksat, Erzincan governorship and TNS Consultancy. It took about 6 months, with an extensive field work, performed by 7 consultants and 5 local employees. Firstly, a model was established by the consultants and then a field work lasting for about 4 months had been carried out. In this field work, 28 local government agencies were visited and all services of those agencies were analyzed through formal data collection mechanisms and modeled accordingly. The total number of services modeled is 950. Later, the model was implemented on Software AG's CentraSite program development environment, and all model parameters for each instance of service inventory were transferred into the electronic environment.

In the project management of EPP, budget, human resource, and time management techniques were integrated to provide a concrete and controlled management environment. At the beginning of the project, the purpose, scope of the project and methods which were going to be used in the project stated explicitly and management accepted them for achieving the expected results.

The main steps of the project were:

1. Project Orientation,
2. Development of field work application model,
3. Formation of notation for business analysis activities,
4. Training of the field work employees for standardized data collection,

5. Data Collecting for model development in Erzincan Province,
6. Development of the Ontological Model and the software development, using CentraSite software platform,
7. Transferring the field work data to the CentraSite,
8. In-depth analysis of the selected three public services, using predefined notation,
9. Integration of these services to electronic servicing environment (www.turkiye.gov.tr), and
10. Preparation of project report and closure.

The analysis and model building phases were divided into ten categories:

1. Analysis of taxonomic structure of government legislation.
   In this step all government legislations were analyzed and categorized. They were ranked according to their priorities in general public management. For example, a law accepted by National Assembly dominates a bylaw and a by-law dominates a mandate. Those relations were listed and categorized in order to build the taxonomic relations.
2. Analysis of hierarchical structure of public institutions.
   In this step, a general hierarchical structure of public institutions was analyzed and categorized according to their rank in public management. This provided a solid base for the work flow of public services.
3. Analysis and categorization of public services.
   The categories of government services were classified according to service users. This was done in parallel with the approach widely used in e-government theory and applications.

   GC: Services given from Government to Citizens
   GB: Services given from Government to Business
   GG: Services given from Government to Government
   GE: Services given from Government to Public Services

4. Building the ontological model.
   In this step, first version of the generic ontological model was built. This model included:
   - Classes: Group of substances which share the same existence relationship in public management domain,
   - Instances: Elements of classes,
   - Relations: Types of relations among the classes, and
   - Attributes: Descriptors or values that are generally associated with individual classes.

   The first version of the ontological model was a generic model for public management which included all activities from planning to the application phase.
5. Forming questionnaires in parallel with ontological model.
   In this step, questionnaires that would be used in the field work were prepared. This was done in parallel with the ontological model. However, it didn't include all answers needed by the model, mainly due to the inability in obtaining values of some attributes from the field work.
6. Compiling questionnaire data collected from EPP field work:

This phase included:

- Training of local team members about the project, method, and inquiries.
- Forming visiting groups and organizing their assigned public agents.
- Initial visits to the public agents and discussion of the results.
- Revision of the questionnaires.
- Completion of the fieldwork.

7. Programming the Ontological Model and transferring fieldwork data to electronic processing environment.

   In that phase, all data collected from the fieldwork was transferred by using a programmed version of the model in Software AG's CentraSite programming environment

8. Designing a model for a selected public service for the application of the generic ontological model.

   Green card public service was selected in order to validate the generic ontological model. The service responded to all aspects of the model in terms of classes, instances, relations and attributes.

9. Integration of the selected pilot service to electronic servicing environment.

   Using extended Event Driven Process Chain model, and submitting the selected Green Card process model to a Business Process Management engine delivered the running portal structure with almost no direct coding for the application development.

10. Feedback and monitoring of the model.

    The model was revised upon the field work experience.

2 The Ontology of Government Services of Turkey

The developed Ontological Model has 21 classes. In the determination of these classes, the nature of public services, their servicing environment, effects to the public and their representation in strategic management of public agencies were taken into account.

Each one of these classes has its own instances. For example, the service class has 950 instances, which refers to all of the public services captured in the EPP Project. The total number of relations among these 21 classes is 166, and the total number of attributes is 110. Table 1, depicts a brief outline of the ontology developed as the first major step of this study.

Table 1. The Overview of the Ontological Model

Class	# Relations	Description	Key attributes
Service	17	The name of the public service presented to the user	Type of service, medium of application for service, medium of declaration for the result, service channel, the category of service, mean time of service, total number of services per year

Table 1. (*Continued*)

Class	# Relations	Description	Key attributes
Regulation	6	All regulations related with service including traditional (non-legislative) ways of performing service, processes and activities	Types, number, date of acceptance, date of application, date of removal, related legislative article
Process	18	Set of functions/ activities to complete service delivery	Cycle of the process
Entity			
User	3	User of the service (citizen, business, public agency, employee)	Depending on the type of user date of birth, gender, citizenship, residency, date of death type of the business, date of foundation, date of close
Unit	3	User of the service	Type of unit, date of foundation, date of close
Plan	10	Any type of plan including strategic plan which has a relation with service	Date of start, date of end, responsible authority
Activity	12	Step of a process	Mean time to complete, average delay time, maximum and minimum times for completion
Input	6	All sorts of documents and data used during the production of the service	Type of input, main or supplementary, electronic or hard print
Output	5	All sorts of documents and data generated during execution of the process	Type of output, main or supplementary, electronic or hard print
Role_1	9	The role of authority delivering the services	Function and type of role
Role_2	1	The role of applicant demanding the service	Function and type of role
Article of the Plan	4	Any article of any plan that has a relation with the service	The article number, chapter, associated goal, goal indicator, performance indicators, responsible authority
Article of the Regulation	5	Any article of any regulation that has a relation with the service	Type of regulation, number, article number
System	2	The system, either manual or electronic, used in the service delivery	Type of system, accessibility, interaction with other systems

Table 1. (*Continued*)

Class	# Relations	Description	Key attributes
Source	3	The source used in the service delivery	Type (human, IT, information, time, budget, moveable assets, fixed assets and their descriptive attributes such as amount, unit, capability)
Risk	7	Any type of event that will negatively impact service delivery or diminish the benefit of the service	Type of risk, probability of risk, impact of risk, risk mitigation
Control	7	Specific activity to check the compliance (ref. Internal Control)	Type, phases, frequency, place
Critical Success Factor	3	An element that is necessary for a service to achieve the expected result	Type, degree
Result/ Benefit	3	A clearly defined outcome	Type of benefit and result
Indicator	9	A numerical measure of the any value related with service and other or other classes	Type of indicator, minimum, maximum, value

The entities in Table 1 can also be represented as given in Figure 1:

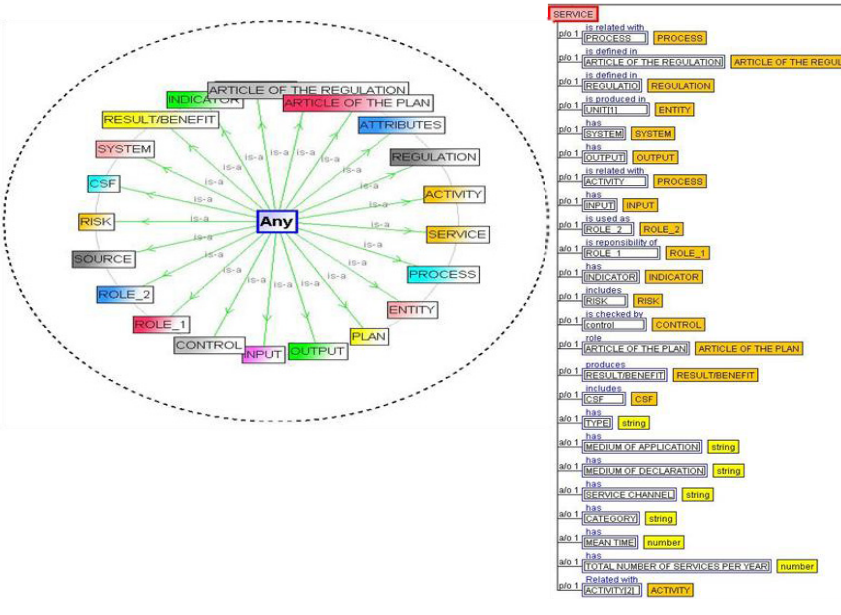


Fig. 1. Classes of the Model

Since the focus is on the service delivery, a closer look at the Service Class would be appropriate. The Service class can be represented as given in Figure 2:

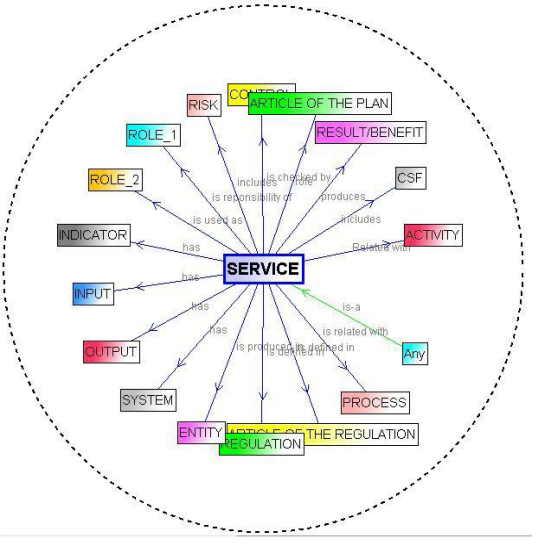


Fig. 2. Service Class and Relations

A further study on the Service class reveals the type of relations with other classes, dimensions and cardinality, as given in Table 2.

Table 2. Tabular View of Service and its Relations

SERVICE	Type of Relation	Class	Relation
Service	is produced in	UNIT	M:N
	is defined in	REGULATION	M:N
	is defined in	ARTICLE OF PLAN	M:N
	Uses	SYSTEM	M:N
	produces	OUTPUT	1:M
	is for	USER	M:N
	is related with	PROCESS	M:N
	is responsibility of	ROLE_1	M:N
	includes	CSF	M:N
	includes	RISK	M:N
	is checked by	CONTROL	M:N
	Uses	RESOURCE	M:N
	Uses	INPUT	M:N
	produces	RESULT/BENEFIT	M:N
	is defined in	ARTICLE OF REGULATION	M:N
	Has	INDICATOR	M:N
	is used as	ROLE_2	

At this stage, the relation between the Service and the process is of M:N type, however, as further refinement occurs, another entity, called Business Process is defined to change the relation to 1:M type. This is in line with the condition that, one or more processes might be required to complete a Service. A Business Process in itself is meant to deliver value to the user, and this is why it will have to be treated as another Service, hence the Service has a cyclic relation with itself; one Service may be a combination of other Services.

One of the important outputs of the EPP Project was the ontology model of the public services inventory. This inventory might provide answers to some fundamental questions for public services in Turkey. Some answers that the inventory provided are:

1. Service to process relation (which processes are required to deliver a particular public service),
2. Information about regulation of any kind (Law, Cabinet decision, etc.)
3. System environment of public services, computerized tasks in the process,
4. Number of inputs and their types, which can also be used in calculating the bureaucracy level indicator,
5. Generated outputs and their types,
6. Total number of services used by citizens, public agencies, businesses and NGO's (non-governmental organizations)
7. The mean time of completion of each service and its activities, cross-linked to the process activities,
8. The role of the applicants,
9. The number of activities in servicing and responsible authorities of these activities,
10. The complexity of business of each service (bureaucracy level indicator) which is scored using number of documents, number of activities and the number of services given per year,
11. The number of usage of any specific type of input for all government services (the frequency of usage of any document and for the high frequency inputs a shared web service would be a fast solution to decrease bureaucracy and duplication),
12. Problems (bottlenecks, redundant activities, organizational issues, and legislative concerns) and solutions / improvements of service delivery, via performing simulations on the process models,
13. Problems and solutions / improvements of service delivery, via online measurement of processes utilizing Business Activity Monitoring,
14. Commitment to delivery (service level agreements) through measuring process performance metrics,
15. Identification of manual vs. computerized operations, and
16. Impact analysis (effect of one perspective on the others). This is basically finding answers to HOW questions, such as How to improve services through process management?; How does technology affect process performance?; How to compare different geographical and cultural approaches on the execution of service delivery processes?; How does a change in a legislation affect service delivery?; How to identify which legislative/regulative changes are required if processes are redesigned?; etc.

Questionnaires were prepared regarding the generic ontological model. As it was stated before, it didn't include all the answers to the model, since it was impossible to get some answers from the fieldwork. The questionnaires were composed of five sections:

1. First part included 8 questions about service to define the fundamental attributes of service, in terms of application and the results of it.
2. The second part was about the legislation framework of service regarding the taxonomic structure.
3. The third part was about inputs (documents in either hard or softcopy) required for application.
4. The forth part was about business flow of service, after application.
5. The last part included outputs of the service and their target users; namely the citizens, businesses, or other public agencies.

3 Business Modeling and Integration

For the pilot implementation of the ontology based process models, three services were chosen, namely Green Card, Retirement, and Traffic Fine Payments.

The Green Card public service is for the citizens who have no social security and inadequate income to personally finance their health expenditures. These citizens in need are issued green cards after 14 step verification through 10 different agencies, entitling them for free medical care. All medical expenses are then charged to the green card, and covered by the government. These citizens constitute 20% of the total population of Turkey. The total health expenditures financed through Green Card system is 3.5 billion USD per annum. Currently, the appraisal of the applicants for the Green Card is a long process which includes collecting information about from 10 different systems by means of paper documentation. It's a time consuming and costly process both for citizens and for the government. Public institutions and application services included in the realization of green service is depicted in Fig 3:

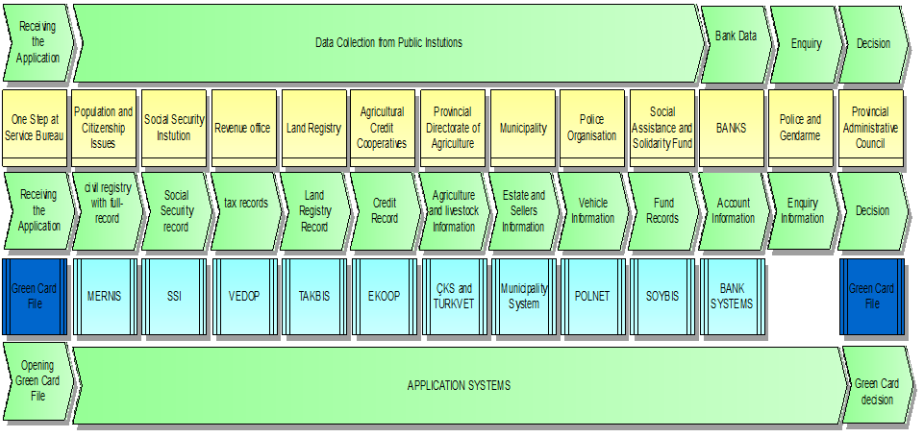


Fig. 3. Value Added Diagram of Green Card Service

Process centricity and continuous process improvement was targeted for the entire service processes covered as part of the project, where Business Process Management (BPM) was used as a tool for the management of the business processes.

BPM Suite used enabled the project team to automate and control processes as they are executed. Process modeling was done to facilitate the use of this suite.

For implementation, WebMethods programming environment was used as the BPM Suite to manage and integrate Green Card Service to electronic servicing environment.

4 Lessons Learnt

EPP project was planned to respond the needs of global modeling of public services. During the modeling efforts, the main problem was about the definition of the service and the process. A clear answer was needed to service and process classes of the model in order to determine the elements of service inventory correctly.

The following answers were developed for the controlled vocabulary:

Service: All individual business processes, combined together to respond a specific needs of the citizen, business or public entities. A service might be started either by the application of user or spontaneously by the bodies of service provider. A value should be delivered by the end of the service which serves a specific need of the user. The value delivered should be for some entity which is out of the body of the service provider. A service can be used by another service.

Process: Complete business process or its subset which is carried out in compliance with a legislation framework. A process can be used by another process.

The main difference between service and process is that, in the service case a value or benefit should be delivered for any entity which is out of the body of the service provider. However in the process case, this is not a must.

Another key issue in the development of the ontological model was the role and relationship concept in the determination of classes, relations and attributes. For example, a public service unit that executes service is a class or an attribute of the service?

In ontological modeling, John Sowa [5] introduces the *firstness* and the *secondness* of concepts. The former is roughly defined as a concept which can be defined without mentioning other concepts. Examples include ion, a man, a tree, etc. The latter is roughly defined as a concept which cannot be defined without mentioning other concepts. Examples include wife, husband, student, child, etc. [6] Since, a public service unit can be defined as an independent concept it should be treated as firstness group. A concept in the firstness group is a class and a concept in secondness group should be an attribute of a class. For example, date of issue of a law is a secondness concept and it should be an attribute of regulation class.

5 Conclusion and Future Work

In this paper, firstly a generic ontological model of the public services was discussed. Then an application method was discussed for a fieldwork study which was realized in order to build business inventory of public services. The application method includes

all the steps from building theoretical ontological model to integration to e- servicing environment.

Our study provides extensive theoretical information and application experience for public modeling analysis and integration to e-government.

Our generic ontological model combines and harmonizes various proven techniques such as qualitative analysis, ontology development, process management including modeling via process chain diagrams, execution and activity monitoring via software support. The model completes a streamlined and integrated cycle from conceptual design through to continuous monitoring and evaluation based on ontology and process approach.

We have also clearly identified the relations and differences between the foundational concepts such as service vs. process, and entity vs. attribute within the scope of government e-servicing.

Finally, we have developed a field study method applicable to almost all government agencies, taking into account management and human resources social and psychological success factors, applying empirical data collection to the theoretical model developed as instances.

Hence, our generic ontological model and the streamlined management process were extensively tested, using sample services and revised into a version that is applicable for all public services.

The methodology developed added to the experience of the team members in the use of the methodology in related areas.

This project was one of the first in Turkey combining multiple disciplines, resulting in an applicable model at government service delivery. We also believe that, our approach will also be useful to other countries as well. However, there is still more work to do, as follows:

- Extension of the model by adding a service life cycle and document (paper or electronic) taxonomy
- Extension of the model by adding instances of missing customs and maritime services
- Extension of the model to cover whole of the country, and establishment of a related governance model, including establishment of dynamic management structure involving major stakeholders
- Detailed application of the model to specific service domains such as education, health and public financial management.
- Possible revisions after wide-spread application, which is possible as Turkey now has a legislation for the usage and population of the model to keep it as a live system.

References

[1] State Planning Organization, Turkey Documents / Reports (2006), http://www.bilgitoplumu.gov.tr/Documents/5/Documents/060700_ActionPlan.pdf

[2] IDS Scheer AG. Method ARIS 7.1. PDF (2009)

[3] Smith, B., Welty, C.: Ontology: Towards a New Synthesis. In: Proceedings of the international conference on Formal Ontology in Information Systems, Maine (2001)
[4] Noy, N.F., McGuinness, D.L.: Knowledge Systems Laboratory. Stanford University (2000), http://www-ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness-abstract.html
[5] Sowa, J.F.: Top-level ontological categories. International Journal of Human and Computer Studies 43(5-6), 669–685 (1995)
[6] Kozaki, K., Kitamura, Y., Ikeda, M., Mizoguchi, R.: An Environment for Building/Using Ontologies Based on a Fundamental Consideration of Role and Relationship (2002), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.75.799&rep=rep1&type=pdf

A Service-Oriented View on Business Processes and Supporting Applications

Sebastian Adam, Matthias Naab, and Marcus Trapp

Fraunhofer IESE, Fraunhofer Platz 1, 67663 Kaiserslautern
{sebastian.adam,matthias.naab,marcus.trapp}@iese.fraunhofer.de

Abstract. Even if SOA has received much attention, there is still no common definition of what a SOA is or what a SOA should provide for business. In this paper, we have therefore introduced a conceptual model on service orientation that explains the impact of service orientation on business processes and supporting applications. We consider this consolidation as an essential step for establishing methods for a better business IT alignment as well as more systematic and integrated business and software engineering in the context of service-oriented enterprises.

1 Motivation

Service orientation and especially service-oriented architecture (SOA) [1][2][3][4][5] have received much attention in research and industry. Nevertheless, there is still no common definition of what a SOA is or what a SOA should provide. Rather, many different definitions can be currently found in the literature.

A traditional and still dominant perspective considers SOA just from a technical viewpoint addressing web service, WS-* protocols and the like [6]. A similar but rather conceptual view highlights SOA as an architecture style for managing heterogeneous application landscape [1]. Finally, in the business community, SOA is increasingly considered as an enterprise architecture or even management concept that promises structuring the business in order to remain more flexible.

While all these specialized definitions and views have their eligibility, they tend to neglect that their might be a more holistic view on SOA. Thus, the multitude of existing viewpoints, which originate from a limited perception, makes it currently hard to get a consolidated picture that could enable a more holistic management of business and IT following the service-oriented paradigm.

In this paper, we therefore claim that all different views on SOA should be consolidated and aligned with traditional business process concepts via a service-oriented business process perspective. Such a perspective could facilitate the explanation of how business issues, application landscapes and technical details are related and how one aspect can influence another. We consider this consolidation as an essential step for establishing a better business IT alignment as well as more systematic and integrated business and software engineering in the context of service-oriented enterprises.

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 39–48, 2010.

As a basis for this aim, this paper therefore introduces a conceptual view on services in the context of business processes and supporting applications that clarifies how the notion of service orientation fits to process-related issues both from an organizational and from a technical perspective. To make that happen, we introduce a service classification and explain how different types of services are used in, respectively produced by, business processes. Furthermore, we clarify the different roles involved in the provision and consumption of services, and we enhance the traditional provider-consumer-broker triangle [1] in this regard. Finally, we deal with the question how service orientation coheres with electronic channels such as the internet. An illustrating example in section 3 shows how our conceptualization can be applied in order to express service orientation in real business settings. The paper closes with a discussion on how we are going to use the conceptualization for developing integrated engineering methods for aligning business and software engineering.

Of course, our idea of consolidating the multitude of views on SOA has been inspired by existing work that also goes towards this direction. In the SOA reference architecture of OASIS [7], for instance, a comprehensive model of many SOA aspects has already been given. However, the business process aspects are rarely covered in this model and only services as such are in the focus of attention. Therefore, a further source of inspiration was the model proposed in [8]. It provides a good coverage of business processes and service concepts and introduces a clear separation of business and IT. We complement the ideas introduced there with more concepts related to the roles involved in typical SOA settings and include more detailed information on the relationship of business services and software services. In particular, beyond the views covered in this paper, our entire conceptual model covers further aspects related to engineering questions and the realization for SOA-based IT-systems.

2 Service Orientation in Business Processes

In this section, the impact on service orientation on business processes is discussed according to our conceptual model we developed in this regard. The section is subdivided as follows. In the first sub section, we explain which types of services can be distinguished. In section 2.2., we then present how the different types of services are aligned with traditional business process elements. In section 2.3., different channels over which services can be requested and consumed are introduced. Finally, in section 2.4., we present different service consumer and service provider roles.

2.1 Service Classification

According to our conceptual model (which is presented using UML in the following), a service is basically "*a clearly defined unit of work, which is provided by a provider and consumed by a consumer and bears an immaterial value for the consumer*". However, in research and industry, the term "service" is often used in a homonymic way expressing each kind of value provided from one party to another without distinguishing the purpose of the service or the type of the provider. It is therefore important to differentiate the types of services that have all been covered under the single term "service" so far.

In our conceptual model, we therefore consider a service as an abstract term that is specialized into business services and (specific) software services (see Figure 1). A business services is *"a service provided by a business unit that is of immaterial value for at least one other business unit"*. Transporting goods from one location to another is an example for a business service in the logistics domain. However, business services can also be for internal purposes (hence, provided to in-house business units) and not only external business services that address external consumers.

In contrast to a business service, a software service is *"a service that is provided by a software system and used by the same or another software system"*. Software service remains also an abstract term and is therefore specialized into either interaction service, function service, or infrastructure services depending on its concrete purpose. While an interaction service is *"a software service controlling a long-lasting interaction among software systems and users"*, a function service is *"a software service providing a self-contained functionality, which can be requested from a software system."* What we call function service here is often called simply service, application service, or web service in other terminologies.

A function service can be either atomic or molecular. While an atomic function service is a function service that is not realized by composition of other function services, a molecular function service is a function service that is realized by such a composition.

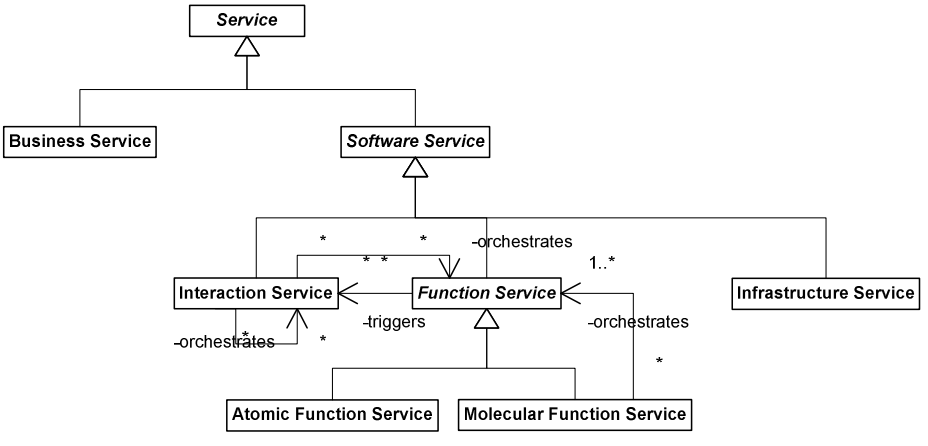


Fig. 1. Service Classification

Function services may trigger interaction services or be orchestrated by them. Hence, both service types have in common that they realize business-specific functionality. However, interaction services are typically different from function services in the way they are consumed. As they control an interaction, they are stateful and need typically a closer integration with the respective service consumer.

In contrast to interaction services and function services, an infrastructure service is *"a software service that does not realize business functionality but technical aspects of a software system"*. Infrastructure services are often not defined and accessible in

the same way as function services with a dedicated interface providing a well-defined functionality. Rather, they often need a very specific integration into the overall technical platform and come with their own ways of interaction. Examples of infrastructure services are process orchestration, data management, data transformation, etc.

2.2 Business Service Alignment

As both business services and software services have a strong impact on the design of modern enterprise architectures, the explanation of how services are related with business process elements is a central purpose of our conceptual model. However, for understanding how the service concept fits to business processes, formalizing the general nature of business processes is a prerequisite.

According to several definitions, we define a business process as "*a self-contained, ordered sequence of business activities initiated by a defined business event (trigger), with defined input and output, a defined start state, as well as a persistent final state of value*". Hence, the purpose of each business process is the realization of a business service through the execution of one or more business activities (see Figure 2).

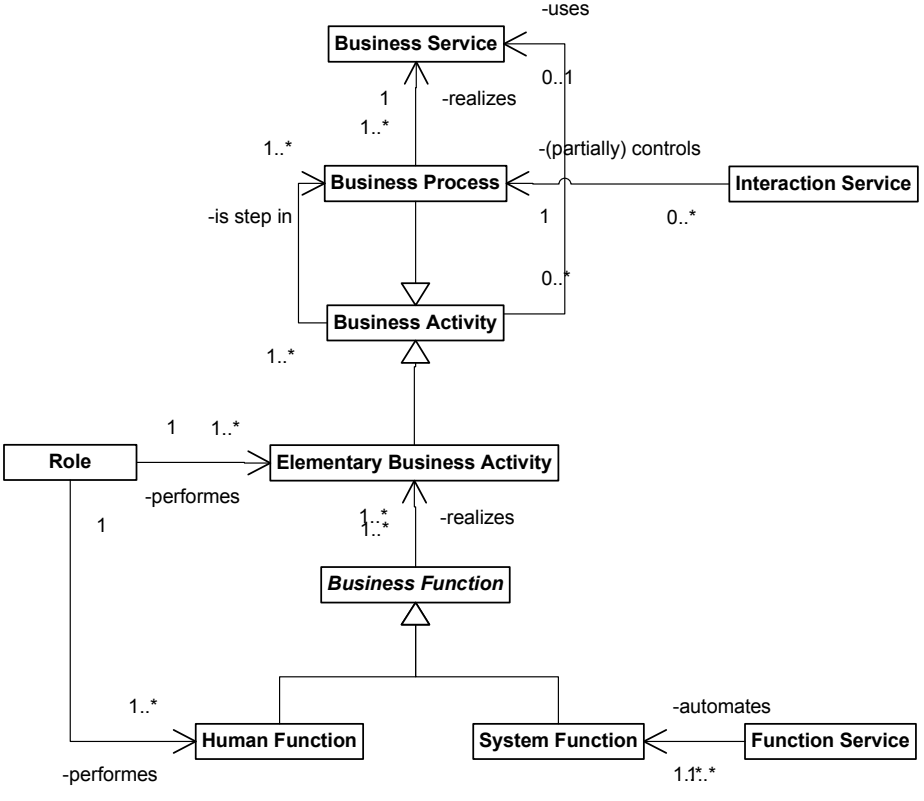


Fig. 2. Service-Business Alignment

A business activity is a step within a business process. However, business activities that are not executed by the organizational unit that provides the process' business service itself will be consumed as business services provided by other organizational units. In the transportation example, for instance, several steps within the transport chain can be outsourced to sub contractors. Hence, these sub contractors provide business services that are used for the realization of business activities in the entire transportation process.

In general, each business activity is either again a business process or an elementary business activity. Thus, each business process consists of at least one elementary business activity. An elementary business activity is *"an atomic step within a business process that ends in a persistent state"*. While "atomic" means that an elementary business unit cannot be further segmented, "persistent" indicates that each elementary business activity has to contribute a persistent value to the business process it belongs to. This means that another business activity is necessary for revocation in case of need. Furthermore, at most one person (acting as exactly one role) is allowed to perform an elementary business activity by interacting with at most one software system.

A business function "*is an atomic step within an elementary business activity*". Thereby, a business function is always executed or performed without any external interruption. Thus, each function is either performed by a human or by system why business functions are classified into human functions and system functions.

A human function is *"a business function that is performed by exactly one person in a specific role as reaction to an external trigger"*. An external trigger may be, for instance, a request from another person, a request from a system, or an environmental context change. In contrast, a system function is *"a business function that is automated by the system as reaction on an external trigger"*. An external trigger in this case may be, for instance, an explicit user request, a call from another external system, or an environmental context change that is detected via sensors.

In the context of a service-oriented application landscape, system functions are provided by function services. In contrast, the purpose of interaction services is mainly the (partial) control or execution of (long-lasting) business processes in which a multitude of humans and function services may be involved.

Services have therefore an impact on business processes on different places. Business services are realized by business processes but may also be consumed by business processes to realize business activities. Software services support the execution of business processes through the provision of corresponding control functionality or the automation of business functions within business activities. However, business services and software services are not only connected in this indirect way. Rather, they can be directly aligned. The next sub section therefore deals with the usage of software services as a channel via which business services can be requested and delivered.

2.3 Service Channels

A central idea behind service orientation has been the possibility to provide and consume business services via function services that are deployed somewhere in the internet. According to this notion, flexible and virtual business collaborations should be enabled. Thus, besides the automation of business functions, function services can also be used to invoke a business service.

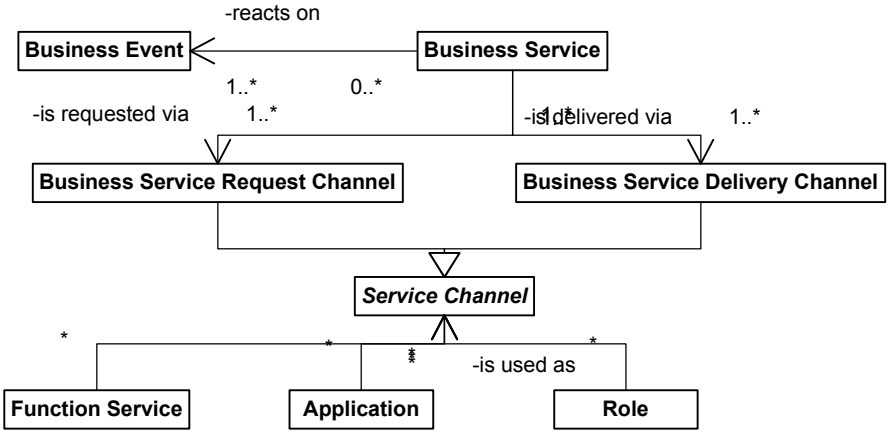


Fig. 3. Channels for Service Request and Service Delivery

In our conceptual model, we have therefore clarified the different channels through which business services can be requested and delivered (see Figure 3). In this regard, a business service delivery channel is *"a channel via which a business service is delivered to the consumer"*. Accordingly, a business service request channel is "*is a channel via which a business service is requested by perceiving a business event via the same channel*". Both the request and the delivery of a business service can be done through different channels. For instance, a (web) application can be used to request / order a product (the receipt of a corresponding order is an example for a business event) while a (human) role is responsible to deliver the ordered product to the customer. In the vision of business process collaborations, however, function services are assumed to be the main request channel, because they allow automating the request of external business services. In particular, business processes on customer side are expected to automatically request business services on provider side for realizing certain business activities through a function service's interface. If the resulting business service is then delivered electronically (via application or via function service) or via a (human) role depends on the type of business service, of course. For instance, real world business services such as transporting goods will be delivered by roles, while stock information will be delivered by function services.

In our model, it is therefore important to clearly distinguish business services and software services even if both may (especially in the future) be strongly aligned. The realization of a business services will always remain a business process, even if software services may be used to request, deliver, or even realize the business service. Nevertheless, the concept of channels allows making explicitly clear how software services and business services are related. Thus, many misconceptions on SOA can be avoided when taking this clarification into consideration.

2.4 Service Provider and Service Consumer

The traditional roles in SOA distinguish service provider, service consumer and service broker. While service provider and service consumer fit to all types of services

introduce in our conceptual model above, service broker are rather relevant for software services only.

In the context of business services, we therefore propose distinguishing the business roles of customer, solution provider, and supplier (see Figure 4). Basically, a business *role "is a set of responsibilities, rights, duties and tasks that can be taken by an organizational unit"*.

A customer is a business role that only consumes business services. Customers never provide a business service within the overall scope of interest (respectively the area of business to be managed). Thus, customers are always the final consumers of business services. This means that a customer is at the upper end of the value chain and all business services provided by the customer are out of scope for the business to be modeled. For example, when modeling the business of a logistic enterprise, the business services of the logistic enterprise's customers (e.g., an automotive manufacturer) would be out of scope.

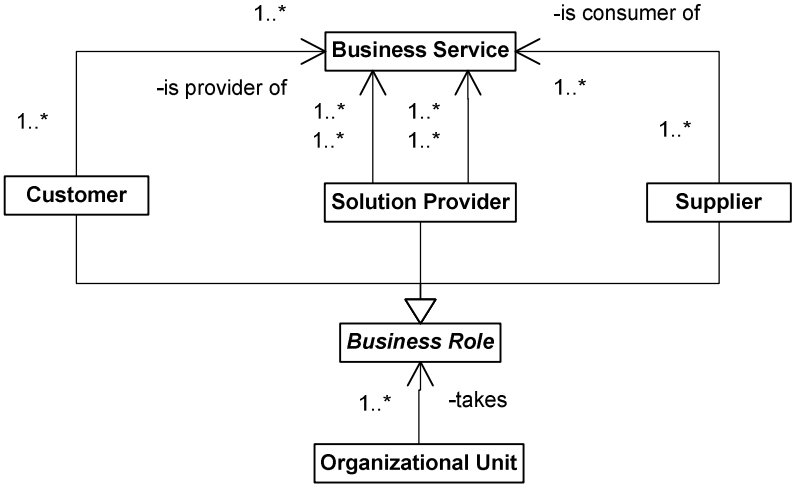


Fig. 4. Business Roles with regard to a certain Business Service

A solution provider, in contrast, is a business role that consumes and provides business services within the overall scope of interest. However, the solution provider only needs the consumed services for providing his own services and not for his own sake. In our transportation example (see section 3), the organization that manages the entire transport from a location A to a location B takes the role of a solution provider, even if transportation services of sub contractors are consumed for some sections in the transport chain.

Finally, a supplier is a business role that only provides business services and never consumes business services within the overall scope of interest. Thus, suppliers are always only providers of business services. This means that a supplier is at the lower end of the value chain in scope and the business services consumed by him are out of scope for the business to be modeled. When modeling the business of a logistics

enterprise, as mentioned above, the business services consumed by the logistics enterprise's sub contractor (e.g., a medium-sized shipping company) would be out of scope.

Through this conceptualization, the value chains that are supported by software services can be expressed in a holistic and uniform manner. Our aim to better integrate business and software engineering in the context of service-oriented enterprises can thereby be supported.

3 Example

To illustrate the applicability of our conceptual model, the previously given examples are briefly summarized, integrated and enhanced into a common example in this section. In Figure 5, the elements used in the example and their relationships are graphically shown. Business services are highlighted through light-gray boxes, while software services are highlighted through dark-gray boxes. The boxes with the dotted background represent elements of the business process hierarchy. The different roles an organization has are expressed through the lines. Solid lines depict organizational units acting as a solution provider. Dashed lines depict organizational units acting as a supplier. Finally, dotted lines represent organizational units that act as a customer within the overall scope of interest. However, it has to be noted that a suitable representation when instantiating our conceptual model is still future work. So far, we have just instantiated the conceptual model in terms of UML objects in order to provide a proof-of-concepts. We are aware that this is not the best way to represent the concepts when modeling real world settings.

Example:
The fictive logistics enterprise LOGISTICS provides a business service "All inclusive Air Freight" in which goods are transported from one location in Europe to another location on another continent (see Figure 5).

For realizing this service, LOGISTICS carries out a complex business process "Air Freight Transportation Process" that consists of several business activities. As LOGISTCS does neither have own airplanes nor own personal at each place on earth, the business activities "Delivery" and "Flight" are subcontracted to plain business service suppliers such as local shipment companies or cargo airlines. Hence, LOGISTICS consumes business service provided by third parties in order to realize its value-added business service mentioned above. LOGISTICS therefore acts as a solution provider with regard to this service.

The business service customers of LOGISTICS, which consumes the "All inclusive Air Freight" service, include companies from the automotive domain, which have to deliver spare parts to any place on earth. For ordering a transport, the customers can request the "All inclusive Air Freight" business services via a function service that is automatically invoked by the customer's ERP system. When the business event "Order received" is perceived via this function service, a new instance of the business process "Air Freight Transportation Process" is created at LOGISTICS. In particular, the interaction service "Transportation Control" within LOGISTICS' internal transport management system (TMS) is started in order to control the execution of this process instance. This interaction service then orchestrates also function services that

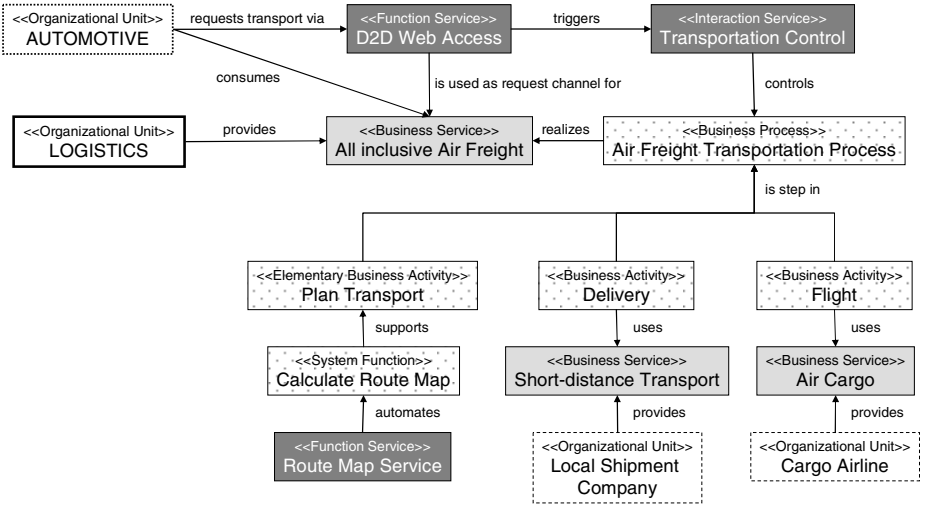


Fig. 5. Visualization of Scenario

act as request channels for the business services provided by the suppliers. Thus, the service-oriented nature of the business process is immediately reflected in a service-oriented IT landscape.

For all (elementary) business activities, which are performed by LOGISTICS itself, appropriate system support exists. As LOGISTICS aims at keeping a high flexibility in their business processes, the own IT landscape is also organized in a service-oriented way. Therefore, each elementary business activity uses system functions provided by (internal) function services. The calculation of a transportation route map is an example for such a system function that is used during the business activity "Plan Transport".

This simple example provides a first insight in the power of our conceptual model. It shows that is quite easy to explain the impact of service orientation in an integrated manner, both from a technical and from a business point of view. In particular, an understanding of the dependencies between different organizational units based on their business service exchanges allows a systematic identifying of appropriate software services support. Hence, representing services in business processes supports assessing the benefits of service-oriented technology in this regard.

4 Conclusion and Outlook

Even if SOA has received much attention, there is still no common definition of what a SOA is or what a SOA should provide. Rather, many different definitions can be currently found in literature. While all these different definitions and views have their eligibility, they tend to neglect that their might be a more holistic view on SOA.

In this paper, we have therefore introduced a consolidated picture on service orientation that is aligned with traditional business process concepts in order to explain

how business issues, application landscapes and technical details are related and how one aspect can influence another one. The views shown in this paper have represented a sub set of our entire conceptual model that also includes many other aspects of SOA and also explicit links to traditional software engineering artifacts.

The purpose of the views presented here is to explain which impacts the service-oriented paradigm may have on business processes as well as on the development of supporting software systems. We consider this consolidation as an essential step for establishing methods for a better business IT alignment as well as more systematic and integrated business and software engineering in the context of service-oriented enterprises.

Currently, we finalize other views of our conceptual model. Furthermore, we have recently started to develop integrated software and business engineering methods that address the systematic alignment of business and IT in service-oriented setting according to our conceptual model. Finally, we are investigating how an instantiation of our conceptual model should be represented.

Acknowledgement

The results of this paper have been developed in the context of the ADiWa project funded by the German Federal Ministry of Education and Research, grant no. 01IA08006F.

References

1. Krafzig, D., Banke, K., Slama, D.: Enterprise SOA: Service-Oriented Architecture Best Practices. Prentice-Hall, Englewood Cliffs (2004)
2. Bieberstein, N., Bose, S., Fiammante, M., Jones, K., Shah, R.: Service-Oriented Architecture Compass. IBM developerWorks (2005)
3. Erl, T.: Service-Oriented Architecture – Concepts, Technology, and Design. Prentice-Hall, Englewood Cliffs (2005)
4. Masak, D.: SOA? Serviceorientierung in Business und Software. Springer, Heidelberg (2007)
5. OASIS: Reference Model for Service-Oriented Architecture 1.0. Committee Specification 1 (2006), http://www.oasis-open.org/committees/download.php/19361/soa-rm-cs.pdf
6. Dostal, W., Jeckle, M., Melzer, I., Zengler, B.: Service-orientierte Architekturen mit Web Services. Elsevier, Amsterdam (2005)
7. OASIS, SOA Reference Architecture, http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-cd-02.pdf
8. Engels, G., et al.: Quasar Enterprise. dpunkt.verlag (2008)

Perspectives for Moving Business Processes into the Cloud

Rainer Schmidt

Aalen University
Anton-Huber-Str. 27
73430 Aalen, Germany
Rainer.Schmidt@htw-aalen.de

Abstract. Moving business processes into the cloud means that the business processes are no longer supported on premise but using a set of cloud services. Cloud services allow to highly reduce upfront investments, change to a "pay-as-you-grow" model and to flexibly react to changes in demand. However, to leverage the benefits of cloud services it is necessary to appropriately integrate the definition of cloud services into business process models. Therefore, three perspectives for defining cloud services are introduced. The functional perspective describes clouds services as the exchange of sub-services between service provider and consumer. The non-functional perspective describes cross-cutting, quality oriented properties of the cloud services. Meta-services as third perspective are used to capture functionality beyond the standard operation of a service, e.g. in the case of service failure.

1 Introduction

Cloud computing is a business transformation based on a number of technical innovations such as virtualization[1], web services [2] and software as a service (SaaS) [3]. There are a number of definitions for cloud computing [4], [5], [6]. However, there is a growing consensus, that the advances of cloud computing can be more easily defined from an economical perspective [7].

Cloud computing may provide public, private or hybrid clouds. Public clouds offer easy to integrate and easy to use cloud services over the internet. In this way, cloud computing transforms the Internet from a source of information to a source of services. By using cloud services enterprises can avoid large upfront investments. E.g. a start-up enterprise is able to minimize its investments because it can start with a low-volume cloud-service consumption and scale up on a pay-for-use only basis. Furthermore, cloud services are elastic. You can increase or decrease their usage according to market requirements without the need to create an infrastructure capable to fulfil the maximum demand. Due to scaling effects, cloud services can be much cheaper than on premise services. Cloud services are designed for internet use; therefore the effort to integrate new cloud services is low. Thus new business processes and thus business models can be implemented quickly and easily and a high agility is achieved.

Up to now, business processes use cloud services in the same way as on premise services. However, there are important differences between on premise services and

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 49–61, 2010.

cloud services requiring the adaptation of business management concepts. Foremost, business processes in the cloud imply an important change in organization. Internal business processes depend upon hierarchic coordination. At the end of the day all participants have a common executive manager being capable to give orders. A business process in the cloud however, does not have such a common manager and thus there is not a hierarchical coordination but a market-based coordination. Disputes have to be solved by negotiations between peers. This implies that contracts have to be agreed upon between the participants of the business process. To do so, it is important to exactly define the business processes and especially the cloud services used. Therefore, this paper will introduce new perspectives for defining cloud services. These new perspectives are the formal foundation for the management of business processes in the cloud.

The paper starts with the presentation of a scenario. A definition of standard perspectives in business processes follows. Then the support of business processes by cloud services is described, and the changes, when moving a business process into the cloud, are analysed. Three new perspectives are introduced: the functional, non-functional and meta-service perspective. Related work is discussed in the following section. Finally a summary and a summary on further work are given.

2 Perspectives for Business Processes

Perspectives represent independently evolving parts of reality. Perspectives are disjoint sets of modelling elements used for process definition. Named aspects, perspectives have been introduced in the workflow [8] and software engineering domain [9]. An approach to unify the usage of the terms in both domains has been done in [10]. There are four core perspectives needed in every process: as shown in the scenario depicted in Figure 1.

It shows a (simplified) business process using the BPMN [11] notation. The business process starts with a request from the customer. He then searches for a book. Assumed he has found a book he enters the delivery address and pays.

The *hierarchical perspective* describes how the service process is composed of sub-processes and activities. In the example of Figure 1, the hierarchical perspective is used to represent the fact, that the business process contains the tasks “Select book“, “Enter delivery address“ and “Pay“.

The *behavioral perspective* defines, when and under which preconditions tasks are performed. The behavioral perspective is often identified with the control flow. The behavioral perspective consists of sequence, gateway elements, etc... Referring to the example above, the behavioral perspective defines, that the process is started by a message of the customer. Upon this message, first the task “Select book“ is performed, than “Enter delivery address“ and final “Pay“.

Furthermore, there is a flow of information between activities. In the *informational perspective* the information that is exchanged between activities is defined. In the example, an data object “Order” is moved from “Select book“ to “Enter delivery address“ and a data object “Completed Order” is moved from “Enter delivery address“ to “Pay“.

The *resource perspective* complements the hierarchical perspective by describing how the activities specified in the behavioral perspective are executed using one or several resources and whether they are operant (active) or operand (passive) resources. Thus it describes the implementation. Resources may be people, organization, information or technology. The resource perspective specifies not only the resource itself, but also the procedure needed to obtain and return the resource, for example from the customer. In the example above, the tasks "Select book", "Enter delivery address" and "Pay" are assigned to the departments Marketing, Logistics and Accounting.

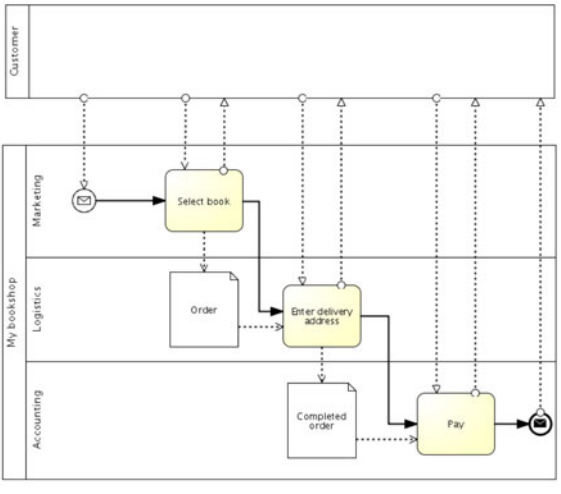


Fig. 1. Perspectives

3 Cloud Services for Business Process Support

The term cloud service includes services already known from SOA but also embraces human based services, infrastructure service etc. Services such as software as a service [3] cannot be described as a software interface alone, how it can be done with web services. The support of business processes by cloud services is done on 4 layers as shown in Figure 2. Often higher services are provided by combining low level services. The business process may be encapsulated as business service.

Business services are services which directly support business processes [12]. They are the interface between the business oriented view of the business process and the more technological view of IT services. They are necessary to aggregate the service properties of underlying services in a way that allows evaluating their appropriateness for business process support. Business services can be either software-based, human-based or a mixture of both. An example is call-centre service provided by a service provider.

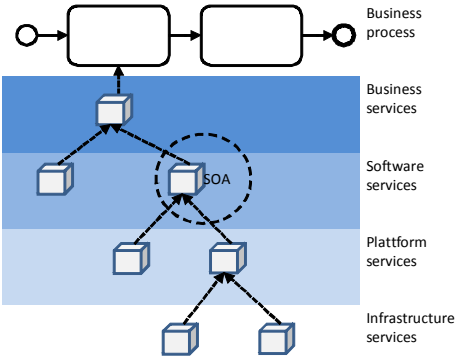


Fig. 2. Layers Cloud Services

Software services exist as two types. There are human-oriented applications which are provided as Software as a Service. And there are application services which are part of so-called Service-Oriented-Architectures [13] that are a popular paradigm for creating enterprise software [14]. A service in the context of SOA is a special kind of interface for an encapsulated unit of software [13].

Platform Services provide support of the development of applications. They provide services for the execution of applications, middleware stacks, web servers etc.

Infrastructure services are more hardware flavoured services which are provided using computers. They may have a human addressee but contain many infrastructure services such as providing computing power, storage etc. They are an important topic in management and practice collections such as ITILV3 [15] or standards such as ISO/IEC 20000 [16] having gained a high popularity.

4 Moving Business Processes into the Cloud

In the good old "new economy" world, the implementation of the business process introduced above would have required large investments in hard- and software. Especially you had to cope with the dilemma, either to buy a very powerful infrastructure to be capable to serve also rare peaks in demand, or to reduce your investments and accept times of bad service due to long response times. Because you do not exactly know how demand will develop you have to endanger your company either by making too huge investments or frustrating customers due to bad service.

Using cloud computing this dilemma can be avoided: You can create your e-shop nearly without upfront hard- and software investments by using cloud-services. Because the e-shop is hosted by a cloud computing provider you can easily enhance your capacities if your business grows. You only pay the performance you actually need ("pay-as-you-grow") and thus avoid either too big or too small investments.

It can be easily seen that using cloud services implies that the business process hitherto confined to the e-shop now includes a number of service providers. However, these service providers do not belong to the same legal entity but are independent, as shown by using choreography instead of orchestration in accordance with the BPMN notation (see Figure 3). The catalogue with the books is now maintained by a

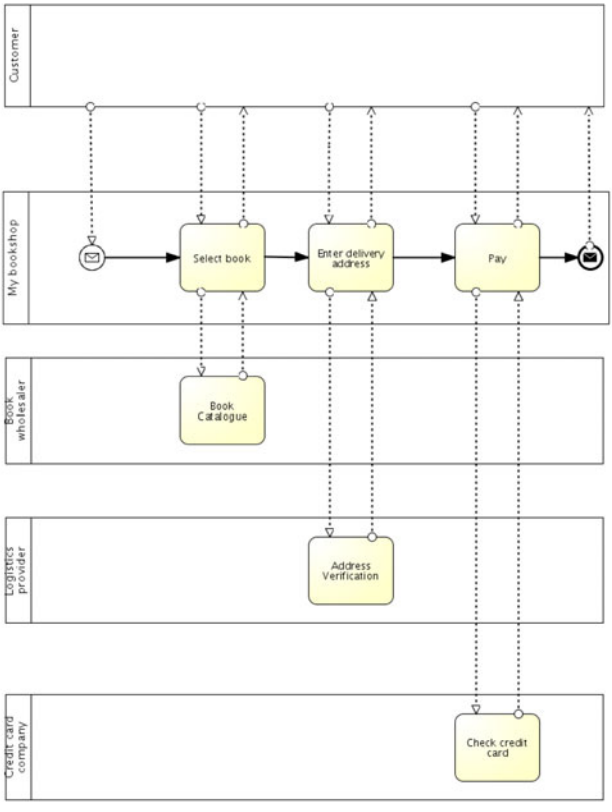


Fig. 3. E-Shop Business Process in the Cloud

wholesaler for books. After the customer has found a book, he enters his address. This address is verified by the logistics provider. Also an expected delivery date is calculated by the logistics provider. Finally the customer enters his credit card number that is checked by the credit card company and to perform the payment.

Due to the distribution of cloud services to different legal and organisational entities it is necessary to describe in more detail, rigorously and explicitly the clouds services used during service execution. Within an enterprise the informal description of a service may be enough to allow an appropriate cooperation of service provider. This is supported by the assumptions that both use the same terminology and thus have a common understanding. However, when service provider and consumer originate from different organizations, such an implicit and mutual understanding of service provider and consumer cannot be assumed. Therefore additional perspectives have to be added to existing business process models. They allow centralizing the information necessary to use cloud services and avoid redundancies. Furthermore, the use of additional perspectives allows improving the separation of concerns by separating hitherto cross-cutting concerns.

5 Perspectives for Defining Cloud Services

To completely define a cloud service it is necessary to not only define the functional properties of the service itself, but also its non-functional properties. Both perspectives are orthogonal, the same service can be provided with different service levels. E.g. the same help desk service can be available from 8 to 8 or around the clock.

However there is still a gap, because these two perspectives consider the operational level only. Therefore a further perspective is required, the meta-service perspective. It contains service acting upon the service not-only in case of failure but also to adapt it to changing requirements etc.

Only the combination of all three perspectives (see Figure 4) allows describing a service completely and evaluating the value of a service. If you define a service but not its availability, you never can rely on the service. If you furthermore define the availability of the service but no means to enforce the agreed upon level of availability, the value of the service is nil.

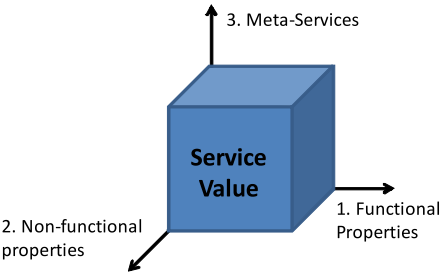


Fig. 4. Perspectives for defining cloud services

5.1 The Functional Perspective

To define the functional properties of cloud services it is necessary to look at the definition of service. There is a long history of service definitions; often the IHIP criteria are used [17]. They define a service by being intangible, heterogeneous, inseparable and perishable. However this definition has been questioned increasingly [18] because there are services not fulfilling the IHIP-criteria. E.g. many mass-produced or industrialized services are highly homogeneous. Furthermore, the IHIP-criteria are also questioned on a formal level, because they define service by the absences of features but not by the presence of features.

Due to the increasing criticism of the IHIP criteria, alternative service definitions have been created such as the one of SD-Logic. Following Service-dominant logic a "service is defined as the application of specialized competences (knowledge and skills) for the benefit of another entity, rather than the production of units of output" [19]. The benefit for the customer should be in the center of interest not the output of production.

Furthermore, there is a shift in understanding from service as an unidirectional activity to a bidirectional one [20]. The provisioning of services is rarely an unidirectional activity of the service provider. In most cases a service is rendered with the help of the

service consumer. That means, also the service consumer has to provide resources etc. to make the provisioning of the service possible. Such an active service consumer is also called prosumer [21]. E.g. given, if the e-book-store wants to use the credit card payment service of the credit company, it has to provide information to the credit card company to enable it to provide the service. If the e-book-store fails to deliver the data appropriately, the credit card is freed from the obligation to check the credit card data. The credit card company may also use subordinated services such as a check of the credit rating of the client.

To reflect this bidirectional character of service provisioning, both the service provider and the service prosumer should be abstracted as a so-called service-system. Maglio et al. [22] define a service system "as an open system capable of improving the state of another system through sharing or applying its resources and capable of improving its own state by acquiring external resources". A service system is defined [23] "as a value co-production configuration of people, technology, other internal and external service systems, and shared information (such as language, processes, metrics, prices, policies, and laws)". The resources shared, applied or acquired by service systems may be divided into four basic classes, namely people, organizations, information and technology [24]. Based on these resources, the service system provide one to many sub-services, by integrating and coordinating resources [25] and co-creates the service desired with other service systems. Thus, there is no service provider producing services in isolation, but service is always the common effort of two or more service systems [26].

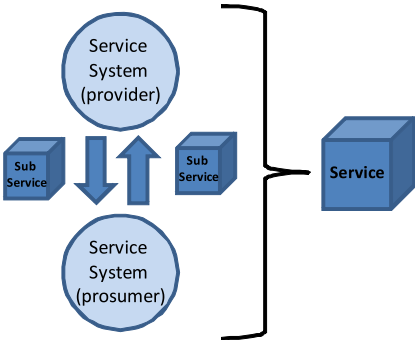


Fig. 5. Functional Perspective of Cloud Services

Based on these fundamental considerations, the functional perspective of cloud services should be represented as a coordinated exchange of sub-services between two service systems representing service provider and consumer. It has to include both the contributed services of the service provider and the service consumer. The sub-services to be exchanged create a system of mutual obligations.

5.2 The Non-functional Perspective

The non-functional perspective describes the non-functional properties of cloud services such as quality, availability etc. Non-functional properties have been identified

as important issue in information systems design (see e.g. [27], [28]). They cannot be assigned to a single entity within a service, they are determined by the service as a whole. Non-functional properties represent cross-cutting issues. They are an important topic for services as shown by the research of O'Sullivan [29]. Following O'Sullivan's point of view, non-functional properties are constraints associated to service functionality. In [29] nine different aspects of non-functional properties are identified: availability, price, payment, discounts and penalties, rights, quality, security and trust.

Availability defines the time and/or location when or where a service can be requested and received by a customer. The prices of a service may depend upon amount of contributed sub-services by the service prosumer. The modalities of payment are agreed upon in the beginning of a service relationship. There may be discounts depending on terms of payment, attributes of the service prosumer or violations of the agreed upon level of service. Also penalties for the failure to fulfill obligation may be part of the non-functional properties. Furthermore quality attributes of the service may be defined. Finally also security and trust are important non-functional properties especially in the context of cloud services. The cloud service provider may receive important information which has to be protected. Furthermore he may use sub-service-providers which have to be included into a sphere of trust.

5.3 The Meta-service Perspective

The meta-service perspective defines services acting upon services. Meta-services are used to capture functionality beyond the standard operation of a service, e.g. in the case of service failure. There are two types of meta-services. First, there are meta-services changing the status of the service. E.g. a service is put into production and changes its status from defined to deployed. In general, a service may have the status undefined, defined, deployed and retired (see Figure 6). When being deployed, a service instance may be instantiated and flagged. Based on these stati, the following meta-services can be defined. The **define**-meta-service creates the necessary definitions of a service. By **deploying**, a service changes from the status defined into deployed. In practice, this is associated with the assignment of resources according to service level agreements. Finally a service may be moved in the status retired by the **retire**-meta-service. When being in status deployed, a service may be **instantiated** that means instances of the service are created. If there is any abnormality, the service instance may be **tagged**. E.g. the instance may be **tagged** if it not performing properly. If the operation of the service instance is back to normal, it is untagged.

The second type of meta-services is those creating or modifying meta-data of a service. Such a meta-service is the **evaluate** meta-service. It is used to collect data about the degree of fulfilment of functional- and non-functional properties defined in order to fulfil requirements. Using the **modify-**meta-service, a service is adopted to changed requirements, either functional or non-functional.

The set of meta-services available to the service consumer may be determined by the business model. E.g a service provider concentrating on standardized, industrialized services may only offer the instantiate and tag meta-services.

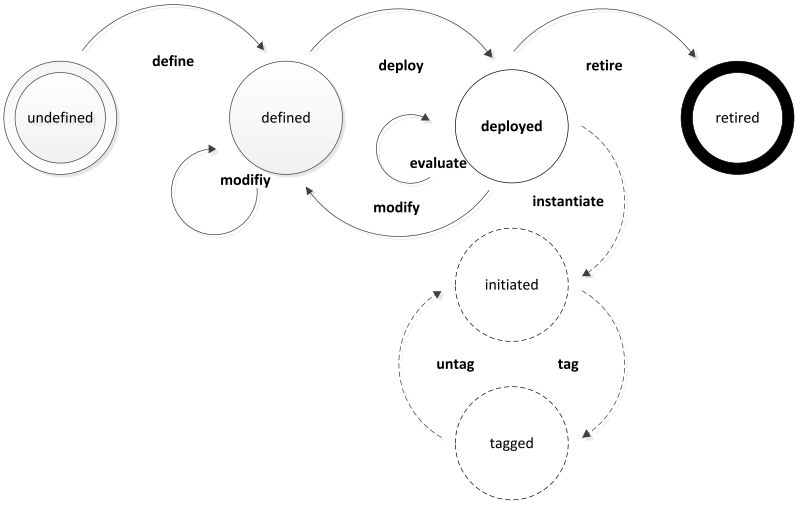


Fig. 6. Service stati and meta-services

6 Example

Now the perspectives shall be used to specify a cloud service for credit payment. The functional perspective of the card payment consists of two interacting sub-services. First, the services prosumer provides the credit-card number and the amount to be paid. Then the credit card company verifies the credit card number and confirms the payment if everything is ok. The non-functional perspective may define that the credit card payment service may be available 24 hours 365 days a year. Furthermore, there may be specified, that there is a maximum interruption of service of 15 minutes in length. The total length of service interruptions may not exceed 4 hours a year. In the meta-service perspective, there may be specified, that there have to be **instantiate-** and **tag-**meta-services. The instantiate meta-service starts the payment process. The tag meta-service is used to indicate service failures.

7 Related Work

There are only cursory approaches for representing services with business process models such as ARIS [30] or BPMN [11]. ARIS allows to model services provided, but neither SLAs nor meta-services. In BPMN 2.0 there are tasks to represent software services. However, there are neither SLAs nor meta-services.

There are a number of approaches using the term meta-service; however, there is no common understanding of meta-services. In [31] the term meta-services is used for the monitoring, optimizing and adjusting of resource services in a grid environment. Meta-services as a kind of intermediary are found in in [32]. Here they “map an existing grid workflow to a service by overriding attributes of the grid workflow”. The term meta-service discovery in [33] or more exactly meta-(service-discovery), thus

meta refers to discovery of services but not to services. In [34] a similar meta-(service-discovery) approach is described.

Meta-services processing meta-data are proposed in [35] those managing meta-data in [36]. Meta-services as a kind of abstract service identifying "a set of services which have similar function" are described in [37]. Thus they are not services acting upon other services. A single service for the composition of e-services is classified as a meta-service in [38]. However, no further meta-services or even a complete framework is identified. The same applies for meta-services in [39], too. Meta-services are used as synonym for certain non-functional properties of services in [40]. Also in [41] QoS provisioning and maintenance are identified as meta-services.

In frameworks like Cobit [42] and ITIL[43] a number of procedures and processes are defined in order to manage services. However, these procedures are neither identified nor managed as meta-service. Instead, an asymmetric handling is introduced, with services as "first-order-objects" and procedures and processes as "second-order-objects"

Service management approaches such as ITIL are organized in an on-premise services provisioning approach. That means ITIL is organized in a way influenced by the idea that all services are provided on premise and not in a outsourced manner. ITIL does not differentiate services whether they are in contact with the customer but only according to their positioning in the service lifecycle. There is no distinction between services interfacing with the customer and those which do not.

Another deficit of approaches such as ITIL or ISO 20000 is that they handle internal and external partners equally. However, external partners are independent legal entities and all interactions with them have to be documented in a very extensive way to be prepared for later juridical disputes. Therefore, processes interacting with external partners have to be designed in another way than processes interacting with internal partners. They have to bear in mind the possibility of a later legal dispute and thus document all interactions.

8 Summary and Outlook

Cloud computing is an increasingly important way to support business processes. However it is necessary to augment business processes by cloud management perspectives to successfully put them into the cloud. Cloud services are externally provided; therefore the required functional and non-functional properties have to be clearly defined. The interfaces between the cloud services and the business processes have to be described in an unambiguous way. This is necessary, because outsourced business processes are in a separate legal entity and therefore a contract has to be agreed upon. Furthermore it is necessary to define meta-services handling the status transition of the services and modifications of the service and its meta-data. The combination of functional, non-functional and meta-service perspective describes a service completely and allows evaluating the value of a service. Especially the meta-services allow describing a service during not only in normal operation, but also in non-standard situations such as service failure.

The identification of perspectives for managing cloud services is the foundation for a number of future areas of research: The meta-services for interfacing between service provider and consumer are only one class of meta-services. There are other classes which may be of importance for the internal management of service systems.

One class of such meta-services are "mirror" meta-services which complement the external meta-services in order to provide services. E.g. the ticket meta-process is mirrored by a process responsible for investigating the deeper causes of service level violations. Contrary to the ticket meta-process its goal is not to re-establish service provisioning as quick as possible but to rigorously identify the root causes of the events. Another class of meta-services are composition and resource-oriented meta-services. They allow to combine existing services or to use resources to provide them. Furthermore, the formal specification of services can be improved.

Because meta-services are likewise services, there are also assigned functional properties, non-functional properties and meta-services. An example for such meta-meta-services is the possibility to complain about the delayed processing of a complaint. Many helpdesks have defined response times and offer escalation mechanisms if tickets are not handled adequately. Of course it is possible to escalate even a level further if this complaint about the complaint handling is not handled properly. Thus, a recursive structure of meta-services is created. Another area of work is the integration of the perspectives into business process modelling methods such as BPMN [11].

Acknowledgements

I would like to thank my colleagues at KSRI during my sabbatical and Gerald Münzl form IBM Germany for the many fruitful discussions.

References

[1] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: Proceedings of the nineteenth ACM symposium on Operating systems principles, p. 177. ACM, New York (2003)

[2] Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S.: Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. IEEE Internet computing, 86–93 (2002)

[3] Turner, M., Budgen, D., Brereton, P.: Turning software into a service. Computer 36, 38–44 (2003)

[4] Vaquero, L.M., Rodero-Merino, L., Caceres, J., Lindner, M.: A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review 39, 50–55 (2008)

[5] Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I.: Above the clouds: A Berkeley view of cloud computing. University of California, Berkeley, Tech. Rep. (2009)

[6] Lenk, A., Klems, M., Nimis, J., Tai, S., Karlsruhe, F.Z.I., Sandholm, T.: What's Inside the Cloud? An Architectural Map of the Cloud Landscape

[7] Carr, N.: The big switch: Rewiring the world, from Edison to Google. WW Norton & Company (2009)

[8] Weske, M.: Workflow management systems: Formal foundation, conceptual design, implementation aspects. University of Munster, Germany (2000)

[9] Lopes, C.V., Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J., Irwin, J.: Aspect-oriented programming. In: Aksit, M., Matsuoka, S. (eds.) ECOOP 1997. LNCS, vol. 1241, pp. 220–242. Springer, Heidelberg (1997)
[10] Schmidt, R., Assmann, U.: Extending Aspect-Oriented-Programming in order to flexibly support Workflows. In: Proceedings of the ICSE 1998 AOP Workshop, pp. 41–46 (1998)
[11] BPMN 2.0 draft
[12] Sanz, J.L.C., Nayak, N., Becker, V.: Business Services as a New Operational Model for Enterprises and Ecosystems. In: Proceedings of the 8th IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE 2006), pp. 61–65 (2006)
[13] Papazoglou, M.P., Heuvel, W.: Service oriented architectures: approaches, technologies and research issues. The VLDB Journal 16, 389–415 (2007)
[14] OASIS, Reference Model for Service Oriented Architecture 1.0 (August 2006)
[15] Ogc, Itil Lifecycle Publication Suite, Version 3: Continual Service Improvement, Service Operation, Service Strategy, Service Transition, Service Design: Service.. Operation AND Continual Service Improvement, Stationery Office Books (2007)
[16] Bon, J., Jong, A., Wilkinson, J.: IT Service Management: An Introduction, Based on ISO 20000 and ITIL V3 (ITSM Library). Van Haren Publishing (2007)
[17] Edvardsson, B., Gustafsson, A., Roos, I.: Service portraits in service research: a critical review. International Journal of Service Industry Management 16, 107–121 (2005)
[18] Lovelock, C., Gummesson, E.: Whither services marketing?: In search of a new paradigm and fresh perspectives. Journal of Service Research 7, 20 (2004)
[19] Vargo, S.L., Lusch, R.F.: Evolving to a New Dominant Logic for Marketing, Juni (2005)
[20] Lusch, R.F., Vargo, S.L., Wessels, G.: Toward a conceptual foundation for service science: Contributions from service-dominant logic. IBM Systems Journal 47, 5 (2008)
[21] Tapscott, D., Williams, A.D.: Wikinomics: How Mass Collaboration Changes Everything, Portfolio (2006)
[22] Maglio, P., Vargo, S., Caswell, N., Spohrer, J.: The service system is the basic abstraction of service science. Information Systems and E-Business Management 7, 395–406 (2009)
[23] Spohrer, J., Maglio, P.P., Bailey, J., Gruhl, D.: Steps Toward a Science of Service Systems. Computer, 71–77 (2007)
[24] Maglio, P., Spohrer, J.: Fundamentals of service science. Journal of the Academy of Marketing Science 36, 18–20 (2008)
[25] Spohrer, J., Anderson, L.C., Pass, N.J., Ager, T., Gruhl, D.: Service Science. Journal of Grid Computing 6, 313–324 (2008)
[26] Grönroos, C.: Service Management and Marketing: A Customer Relationship Management Approach. Wiley, Chichester (2000)
[27] Glinz, M.: On non-functional requirements. In: Proc. RE, vol. 7, pp. 21–26 (2007)
[28] Chung, L.: Representation and utilization of non-functional requirements for information system design. In: Andersen, R., Solvberg, A., Bubenko Jr., J.A. (eds.) CAiSE 1991. LNCS, vol. 498, pp. 13–15. Springer, Heidelberg (1991)
[29] O'Sullivan, J.J.: Towards a precise understanding of service properties (2006)
[30] Scheer, A.W.: ARIS-business process modeling. Springer, Heidelberg (2000)
[31] Du, Z., Lau, F.C., Wang, C., Lam, W., He, C., Wang, X., Chen, Y., Li, S.: Design of an OGSA-Based MetaService Architecture. In: Jin, H., Pan, Y., Xiao, N., Sun, J. (eds.) GCC 2004. LNCS, vol. 3251, pp. 167–174. Springer, Heidelberg (2004)
[32] Lee, S., Choi, J.: Meta Services: Abstract a Workflow in Computational Grid Environments. In: Sunderam, V.S., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2005. LNCS, vol. 3516, pp. 916–919. Springer, Heidelberg (2005)

[33] Buford, J., Brown, A., Kolberg, M.: Meta service discovery. In: Pervasive Computing and Communications Workshops, Pisa, p. 6 (2006)
[34] Friday, A., Davies, N., Wallbank, N., Catterall, E., Pink, S.: Supporting service discovery, querying and interaction in ubiquitous computing environments. Wireless Networks 10, 631–641 (2004)
[35] Hau, J., Lee, W., Newhouse, S.: Autonomic service adaptation in ICENI using ontological annotation. In: Proceedings of Fourth International Workshop on Grid Computing, 2003, pp. 10–17 (2003)
[36] Hau, J., Lee, W., Newhouse, S.: The ICENI semantic service adaptation framework. In: UK e-Science All Hands Meeting, pp. 79–86 (2003)
[37] Qi, S., Hai-yang, W.: Meta Service in Intelligent Platform of Virtual Travel Agency. In: First International Conference on Semantics, Knowledge and Grid, SKG 2005, p. 116 (2005)
[38] Casati, F., Sayal, M., Shan, M.C.: Developing e-services for composing e-services. In: Dittrich, K.R., Geppert, A., Norrie, M.C. (eds.) CAiSE 2001. LNCS, vol. 2068, pp. 171–186. Springer, Heidelberg (2001)
[39] Crawford, C.H., Bate, G.P., Cherbakov, L., Holley, K., Tsocanos, C.: Toward an on demand service-oriented architecture. IBM Systems Journal 44, 81–107 (2005)
[40] Norman, T., Jennings, N., Faratin, P., Mamdani, A.: Designing and implementing a multi-agent architecture for business process management. In: Jennings, N.R., Wooldridge, M.J., Müller, J.P. (eds.) ECAI-WS 1996 and ATAL 1996. LNCS, vol. 1193, pp. 261–276. Springer, Heidelberg (1997)
[41] Rodrigues, R.F., Soares, L.F.G.: Inter and intra media-object QoS provisioning in adaptive formatters. In: Proceedings of the 2003 ACM symposium on Document engineering, pp. 78–87. ACM, New York (2003)
[42] V.H. Publishing, IT Governance based on Cobit 4.1 - A Management Guide. Van Haren Publishing (2007)
[43] S. Office, The Official Introduction to the ITIL 3 Service Lifecycle. The Stationery Office Ltd., Norwich (2007)

Viewpoints Reconciliation in Services Design: A Model-Driven Approach for Highly Collaborative Environments

Sophie Ramel, Sylvain Kubicki, Alain Vagner, and Lucie Braye

Public Research Centre Henri Tudor
29, avenue J.F. Kennedy, L-1855 Luxembourg-Kirchberg, Luxembourg
{sophie.ramel,sylvain.kubicki,alain.vagner,lucie.braye}@tudor.lu

Abstract. In highly collaborative business contexts, services design is even more important than in other contexts, as services are more complex and involve multiple and composed services. Such contexts are ideal for service composition and thus service reuse. This paper presents an ongoing research project whose goal is to support the design of such services by reconciling different viewpoints, following a model-driven approach. It focuses on the so-called "transactional aspects" of the service design, corresponding to the processes realizing the service and showing services compositions between the involved parties. It is applied to a case study in the construction sector.

Keywords: service design, business process, MDE, UML, BPEL.

1 Introduction

In collaborative business environments, the processes related to the design of services are often ad-hoc, or relying on the know-how of the various actors. Such environments are characteristic of numerous business fields such as the Architecture, Engineering and Construction (A.E.C.) sector. Construction projects involve numerous practitioners, for short durations, in various contractual contexts. Therefore processes cannot really be pre-defined and have to remain flexible enough. Modelling the heterogeneous A.E.C.-dedicated services could enable composing them to answer specific requirements of a collaborative situation (i.e. a construction project). We previously carried out a service-driven innovation process in Luxembourg[1]. This process was composed of open-innovation activities[2] and involved several actors: researchers (both domain experts and IT scientists), standardization body and representatives of the main construction trades. During the experimental stages and the commercial transfer of these services we noticed the importance to adapt these services to the specific context of each construction project in which they are used. This article presents the results of the ongoing Dest2Co project, applied on a case study extracted from A.E.C. The part 2 describes the views proposed in this service design process. The part 3 suggests a design method based on these views. Finally the conclusion opens prospects in terms of tool development and research validation.

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 62–68, 2010.

2 Service Design Views

Our previous innovative services design allowed us to identify several roles involved. In order to organize their specific concerns, we adopted an architectural framework (cf. ISO/IEC 42010) proposing the definition of viewpoints for these roles and organizing the related models into views: the business requirements view, the business solution view and the technical solution view. Each view is in addition linked to the domain model.

This viewpoint approach could be compared to Architectural Frameworks (like Zachman[1], TOGAF[2] and ArchiMate[3]), which organize views on the enterprise in layers. However, these layers help define links between different elements (which can be seen as a decomposition of the overall system into subsystems), while our approach considers that each model is a representation of the same element (the service) from its specific viewpoint. This separation of concerns could be compared to the Model Driven Architecture[4], and their different models (CIM, PIM, PSM). However, our goal is not to design a system but a set of services, and our models are not necessarily based on MOF[5] meta-models. Because of this, this approach should rather be considered as part of Model Driven Engineering (even if we do not plan to automate completely all the model transformations).

The three views are presented in the next paragraphs through a case study related to the design of an intermediary service-based system to share documents in an A.E.C. project. The examples focus on the service(s) provided by this intermediary, enabling to ask partners to review a project's document.

2.1 Business Requirements View (BRV)

The first view in the Dest2Co approach concerns requirements from a business perspective, corresponding to the details of what domain practitioners want to achieve with this services' design project. These can be expressed with high-level concepts, referencing elements of the domain, but they are not sufficiently formalized to require the use of models. In addition, we do not describe transactions nor business processes yet as we are only expressing global requirements.

In our case study, this view allows us to collect general requirements on the sharing of documents, like the use of standards for naming documents, the management of reviews and validations, or the practices of documents' versioning.

2.2 Business Solution View (BSV)

This view consists in the first model concerning the solution (how we can do) instead of the problem (what we want to do) expressed in the BRV. It could be created by business analysts, or even by business experts sensitised in services

[1] http://www.zachmaninternational.com/index.php/the-zachman-framework
[2] http://www.opengroup.org/togaf/
[3] http://www.archimate.org/
[4] http://www.omg.org/mda
[5] http://www.omg.org/mof/

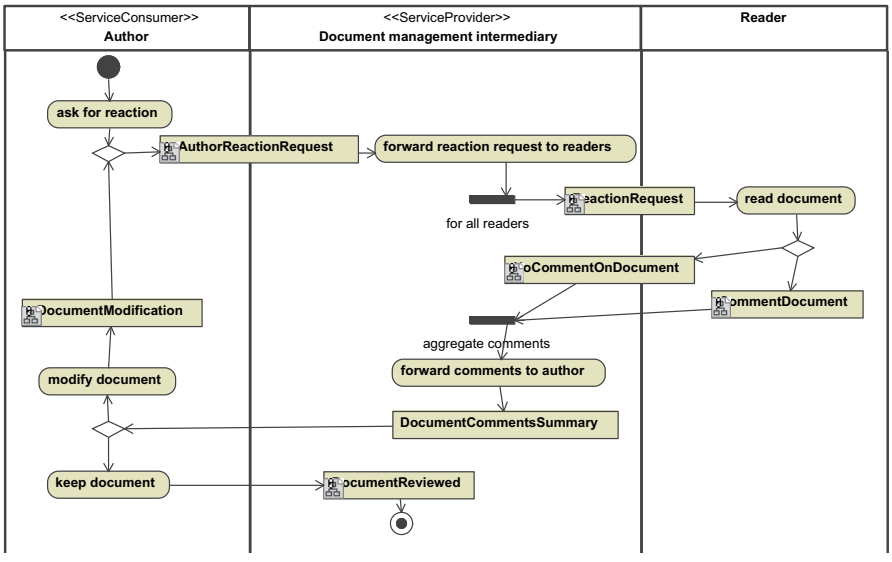


Fig. 1. "Review document" service activity diagram for Business Solution View

modelling. Its goal for transactional aspects is to model the choreography of exchanged messages (and potentially composed services) needed to realize the service, between all the involved partners, from a business point of view. Because this view requires simplicity, we have chosen to use UML diagrams to represent these services' business processes (see details on the use of UML activity diagram to model services in [3]).

Swim lanes (partitions) are used for the different roles involved in the service (service consumer, provider, and third parties). This activity diagram should represent the business activities, and the composed business services' calls (modelled as sub flows), as well as information in the form of objects exchanged between activities. However, it should not contain technical details. In particular it should not contain systems but only human/business roles, and only activities with a business meaning.

In addition to the activity diagram, we propose to model the exchanged messages definitions using class diagrams, based on classes defined in the domain model, by hiding irrelevant attributes/associations.

In our example (see Fig. 1), we have identified the document management intermediary as the service provider, and the document author as the service consumer. Reviewers are additional actors involved in this business service.

2.3 Technical Solution View (TSV)

The Technical Solution View consists in the view of IT experts on the service. It is created based on the Business Solution View by taking into account technical

specificities, choices, architectures, etc. We have divided this view into two levels: a "non-executable level" for a first approach defining the exchanges of technical services but without all the implementation details, and an "executable level" adding the technical details required for the future implementation.

Technical Solution View 1: Non-executable Level. This view is the first one taking into account IT for the design of the service. It corresponds to a software analysis phase, realized by a software analyst, and should thus take into account IT globally, without requiring architectural knowledge. Because it is based on the Business Solution View, we have chosen to keep the UML notation, but to change the way we use it: the process corresponding to the realization of the service is still modelled using an activity diagram, but the swim lanes correspond to software and not human roles, and we added some technical activities and calls to software services. However, some human activities remain stereotyped "humanActivity". Objects now correspond to technical messages sent between two systems: they should thus be refined, based on the higher level messages of the Business Solution View, to add the technical information needed.

In our example (Fig. 2), author and reader roles have been replaced by their IT interfaces. The document management intermediary has been replaced by "DMS" (for Document Management System). Since documents are managed centrally by the DMS, activities called "getDocument" and "store document in system" have been added, and the messages exchanged have been changed.

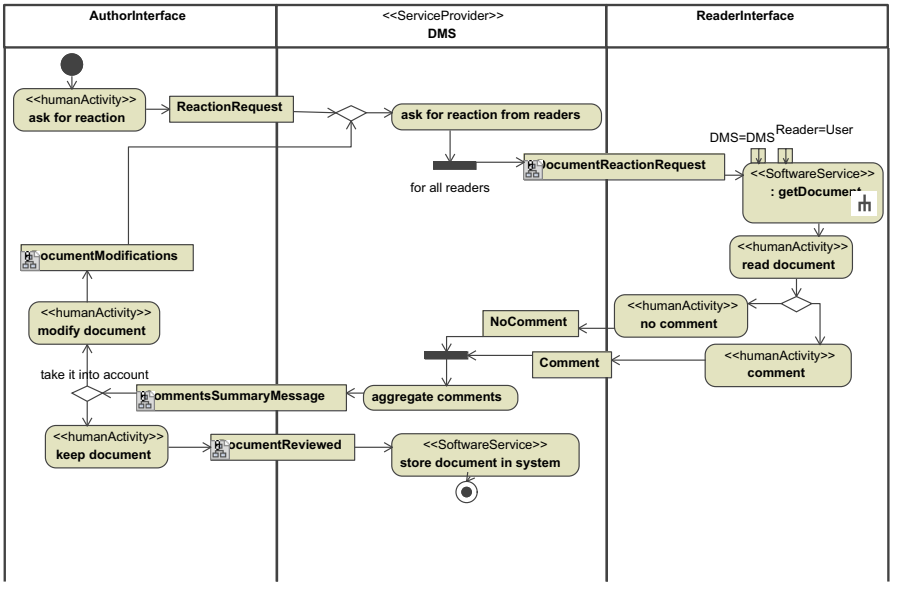


Fig. 2. "Review document" service activity diagram for non-executable TSV

Technical Solution View 2: Executable Level. On the executable level, the design of services consists in refining the service calls into executable processes. As we wanted the resulting design to be executable in different contexts, we have chosen the most popular language for the orchestration of services, BPEL. Messages are then naturally modelled using XML Schemas.

To create this view based on the previous one, UML activity diagrams have to be transformed into BPEL processes, switching from a choreography to an orchestration in a manner similar to [4]. Each swim lane corresponding to services needing to be orchestrated will require a BPEL process. The WSDL files describe the various atomic services. Figure 3 illustrates (using the graphical notation of the Eclipse BPEL Editor) the two BPEL processes needed in our example, one for the DMS and another one for the reviewer interface (which will in fact probably be implemented by the same DMS system, but this should be decided later). Compared to the process from the non-executable level, we added an activity "GetReviewers" to fetch the reviewers for this document. In addition, we converted the human activity "read document" into an invocation "InvokeCommentDocument" to gather the information required in the process response.

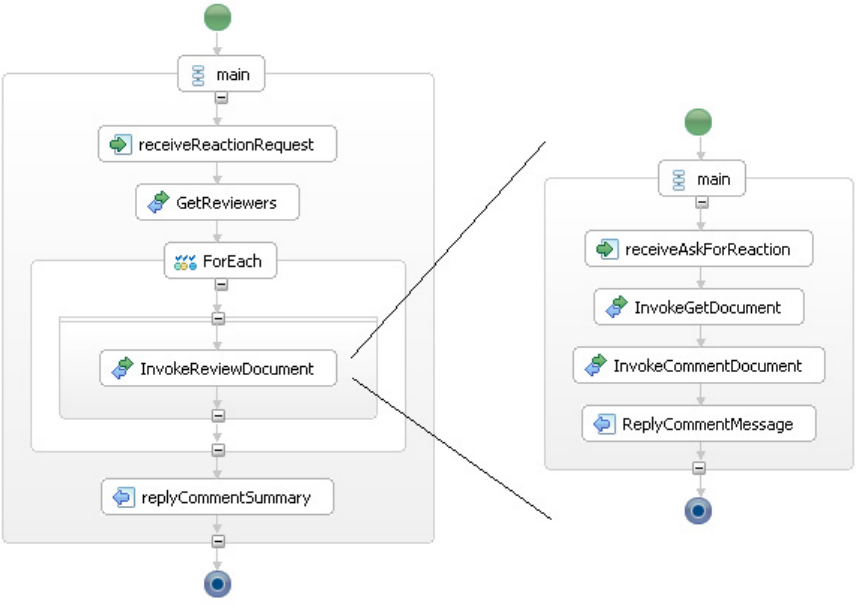


Fig. 3. "Review Document" service BPELs for executable Technical Solution View

3 Towards a Service Design Methodology Based on Viewpoints

The Dest2Co methodology defines how the different actors can design the services for a single project situation by using the different views. It could be

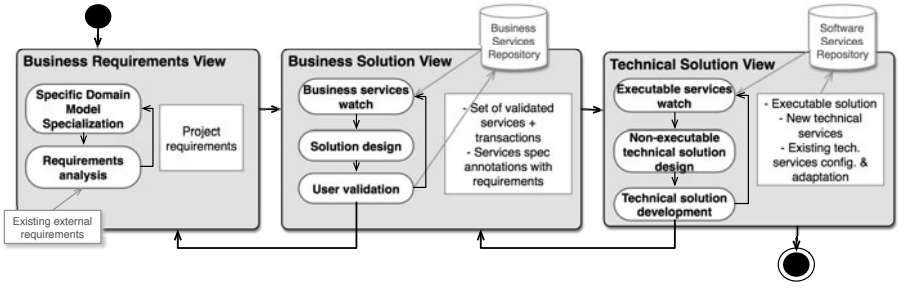


Fig. 4. Dest2Co methodology based on views

considered as a "classical" top-down design process, but at the same time as a bottom-up approach in order to foster service reuse. An overview of this process is shown in Fig. 4. It contains activities that contribute to refining the services and domain models. Moreover, it is noteworthy that the process is mainly iterative, since activities can imply changes in other views.

One of the main goals of this methodology is to enable integrating all different views in one process. The views are refined and reused during the whole process but mainly consist in different facets of the same reality, i.e. services. The integrative aspect of this methodology is based on Model Driven Engineering techniques [5], by defining meta-models for each view, as well as for the business domain, and defining the relations between these meta-models. These relations enable model transformations for passing from one viewpoint to another, and create the structure of models, to be completed by the analyst/designer. Keeping the relations between the models allows taking updates into account later on.

The service repository should enable to search for existing services, and will not be limited to only technical descriptions of services, as is often the case in software services repositories. As a consequence, we think this repository will be helpful for service reuse even on the business level, by fostering business service reuse [6].

The "user validation" step will be based on an innovative approach called "Efficient"[6] [7], that enables validating the models of electronic transactions through an animation tool allowing business experts to "play" the model as if it was already implemented. By animating the transaction, business experts understand the model better and can correct the problems (missing or incorrect information in messages sent, incorrect order of messages, etc.), then animate the transaction again, until they consider the model valid for their requirements. We plan to adapt this methodology and tools to support the service orientation, and use it specifically for the Business Service View.

4 Conclusion

This paper presents the results of an ongoing research project aiming at defining a methodology for the design of services. It is dedicated to highly collaborative environments and supports the reconciliation of viewpoints.

[6] http://efficient.citi.tudor.lu/

The methodology will be partially supported by software tools. We plan here to build these tools by extensively reusing open source components. We have chosen to use the Eclipse platform, because it provides many model-oriented plugins, meta-modelling facilities and tools allowing the creation of plugins. Parts of the transformations between views will be implemented, and editors/wizards will be provided to complete the models. A validation of the methodology will then be performed through the design of existing services in a real A.E.C. project case.

Acknowledgments. This article was supported by the Dest2Co project funded by FNR in Luxembourg.

References

1. Kubicki, S., Dubois, E., Halin, G., Guerriero, A.: Towards a Sustainable Services Innovation in the Construction Sector. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 319–333. Springer, Heidelberg (2009)
2. Chesbrough, H.: The era of open innovation. MIT Sloan Management Review 44(3), 35–41 (2003)
3. Ramel, S., Grandry, E., Dubois, E.: Towards a Design Method Supporting the Alignment between Business and Software Services. In: 33rd Computer Software and Applications Conference, COMPSAC 2009, Seattle, WA, pp. 349–354 (2009)
4. Mendling, J., Hafner, M.: From WS-CDL choreography to BPEL process orchestration. Journal of Enterprise Information Management 21(5), 525–542 (2008)
5. Favre, J.M.: Towards a Basic Theory to Model Driven Engineering. In: Workshop on Software Model Engineering, WISME 2004, Lisboa, Portugal, October 11 (2004)
6. Adam, S., Ünalan, Ö., Riegel, N., Kerkow, D.: IT Capability-Based Business Process Design through Service-Oriented Requirements Engineering. In: Enterprise, Business-Process and Information Systems Modeling. LNBIP, vol. 29, pp. 113–125. Springer, Heidelberg (2009)
7. Mammar, A., Ramel, S., Grégoire, B., Schmitt, M., Guelfi, N.: Efficient: A Toolset for Building Trusted B2B Transactions. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 430–445. Springer, Heidelberg (2005)

Towards Extending BPMN with the Knowledge Dimension

Inese Supulniece, Ligita Businska, and Marite Kirikova

Riga Technical University, 1 Kalku, Riga, LV-1658, Latvia
Inese.Supulniece@rtu.lv, Ligita.Businska@cs.rtu.lv,
Marite.Kirikova@cs.rtu.lv

Abstract. At the root of the success of modeling, design, reengineering, and running business processes is effective use and support of organizational knowledge. Therefore, the relationships between a business process and organizational knowledge should be clearly documented. Several methods have already been elaborated that introduce the process dimension into knowledge management or a knowledge concept into business process modeling. However, the usability of these methods is restricted either by applicability only to knowledge-intensive processes or by relying on sophisticated or uncommon modeling techniques. Building on the experience accumulated by researchers working at the intersection of business process modeling and knowledge management, this paper proposes to extend the well-known Business Process Modeling Notation with the knowledge dimension so that using a common modeling technique it would be possible to relate different forms of knowledge, information and data to the business process model. The approach is demonstrated by a case study from a data base integration project at the Bioinformatics Company.

Keywords: business process modeling, knowledge management, knowledge intensive business process, data, information, knowledge.

1 Introduction

Business process modeling (BPM) has proven its value in achieving more effective performance of enterprises. It is also well-known that a sustainable competitive advantage of an enterprise in changing environment can be enabled by knowledge management (KM) [1] that facilitates identifying, creating, representing, sharing, integrating, and effectively managing different types of knowledge within an organization. BPM and KM have similar purposes and may use some similar techniques for the achievement of these purposes. Therefore there is a possibility to merge these approaches by at least to some extent combining their advantages. Several attempts have already been made to introduce the process dimension into KM or the knowledge concept into BPM, e.g., [2, 3, 4, 5, 6, 7, 8 and 9]. These attempts make it clear that the following issues at the intersection of BPM and KM are to be taken into consideration:

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 69–81, 2010.

- It is necessary to distinguish between knowledge used to perform business processes (BP) and knowledge created as a result of BP activities.
- Information about the process itself is important knowledge for an organization.
- In knowledge-intensive organizations it is necessary to accumulate process related knowledge in order to be able to redesign BPs.
- In modeling of a knowledge-intensive BP it is necessary to present and separate data and information from knowledge.
- In knowledge-intensive BPs it is necessary to separate tacit and explicit knowledge in order to identify which tacit knowledge should be transformed into explicit knowledge, such as documents, rules, systems, etc.
- To utilize the computer system effectively certain knowledge about it is needed to obtain information (not meaningless data) from it, i.e., focusing on how effectively individuals use IT.

Several modeling methods that can partly handle the above-mentioned issues have already been proposed [2, 3, 4, 5, 6 and 7]. However, the usability of these methods is restricted either because of their applicability to knowledge-intensive processes only or sophisticated or uncommon modeling techniques. Thus, the research presented in this paper focuses on the development of widely applicable techniques that can represent KM issues in BP models clearly and with a sufficient level of details.

In this paper, we propose to extend one of the best known modeling techniques, namely, the Business Process Modeling Notation (BPMN) [10, 11 and 19] with the knowledge dimension in order to take into consideration issues relevant at the intersection of KM and BPM on the one hand; and to provide easily applicable means for a knowledge aware process documentation and analysis on the other hand.

BPMN was selected as the notation for process modeling, because it has become a de facto standard for modeling BPs [11]. The proposed extension of BPMN roots in concepts implemented in the Knowledge Modeling and Description Language (KMDL) [6]. KMDL is a process oriented knowledge management method which supports modeling of knowledge intensive processes and a sequence of tasks representing knowledge processes in tasks and knowledge flow between activities [6]. KMDL was selected, because it incorporates requirements for knowledge intensive process modeling better than other techniques [12]. However, KMDL does not distinguish between data and information. Therefore, to introduce this distinction, some of the KMDL concepts were modified before the inclusion of the knowledge dimension into BPMN.

The rest of the paper is structured as follows – Section 2 briefly presents related works relevant to the research problem. Section 3 describes basic concepts for including the knowledge dimension in BP models in general and BPMN in particular. In Section 4 we demonstrate the extended BPMN in the case study from a data base integration project of Bioinformatics Company by using the proposed approach for knowledge intensive process modeling and for routine BP reengineering purposes. The paper concludes with Section 5, where the results and further research are discussed.

2 Related Work

Integration of BPs and knowledge flow has attracted the attention of research communities and has rapidly become a hot research topic. According to a global Delphi

study about the future of KM, integration of KM into BPs was identified as the most pressing as well as the most promising practical and theoretical task in KM [13]. There have been many attempts to integrate KM and process orientation. Generally, different research approaches can be segregated into two different categories: (1) process oriented knowledge management and (2) knowledge oriented BP modeling. Approaches of the first category consider BPs as a subject of KM. Approaches of the second category take BPs as the initial point for KM. Some of the approaches belonging to each category are briefly discussed below.

Process oriented knowledge management:

- Method for modeling knowledge flow using Petri net [2] – describes how knowledge flows through knowledge nodes, thus formalizing dynamics of knowledge flows through computational models. The focus is made on the knowledge dimension, as a result, the process view is not clearly presented.
- Method for modeling knowledge intensive processes (KMDL) [6] – formalizes knowledge intensive processes with a focus on certain knowledge-specific characteristics in order to identify process improvements in these processes. The method is hard to understand and to apply for the purpose of facilitating the involvement of modeling participants.
- Method for integration of KM into BPs (GPO-WM) [5] – describes and evaluates the current state of handling core knowledge domains, to gather improvement ideas for systematic knowledge handling and to integrate selected KM methods and tools into existing BPs. The method does not allow the modeling of knowledge conversions.

Usually BP oriented knowledge management methods focus on storing and sharing knowledge. As a result, they lack the ability in an adequate manner to model the decisions, actions and measures, which are causing a sequence of processes. From a practitioners' point of view, existing approaches require significant effort for analysis, design and implementation. Furthermore, most of these methods are convenient only for knowledge management experts and require additional training for non-experts.

Knowledge oriented BP modeling:

- Method for integrated modeling of BPs and knowledge flow based on a Role Activity diagram (RAD) [2] – provides integration of BPs and knowledge flow and helps KM build on existing process management efforts. However, further research is needed for the analysis of virtual knowledge flow through social networks, and the coordination of BP and knowledge flow redesign.
- Method for service oriented KM (PROMOTE) [7] – integrates strategic planning with the evaluation of knowledge management and PB management and defines knowledge management requirements on the basis of business needs. The method does not explicitly separate tacit and explicit knowledge. Besides, it has some weaknesses in the knowledge process representation.

Knowledge oriented BP modeling methods do not present different types of knowledge conversion, such as internalization, externalization, socialization and combination that are relevant in knowledge intensive processes. In addition, they don't differentiate between tacit and explicit knowledge.

In Table 1 the above-mentioned methods are compared with respect to issues relevant to the intersection of BPM and KM. If the method supports the issue – it is denoted '+'; if it does not support the issue – it is marked '-'; if it partly supports the issue – it is denoted '-/+' in the appropriate cell. The table contains only issues which are targeted in this paper and does not seek a representation of all issues relevant at the intersection of BPM and KM.

Table 1. Comparison between methods applicable at the intersection of BPM and KM

	Based on Petri net	KMDL	GPO-WM	PROMOTE	Based on RAD
Tacit and explicit knowledge can be modeled	-	+	+	-	+
Owner of knowledge and location where knowledge can be obtained can be clearly stated	+	+	+	-	+
Knowledge conversion types can be specified	-	+	-	+	-
Knowledge used to perform BPs and knowledge created as a result of BP activities can be separated	-	+	-	-	-
Data and information can be separated from knowledge	-	-/+	-/+	-	-
Presents knowledge about computer system (needed to obtain information from it)	-	-/+	-/+	-	-

Table 1 shows that none of the methods can clearly distinguish data and information that form a basis for knowledge creation and sharing. It also shows that KMDL covers most issues targeted by this paper. Therefore KMDL is taken as a basis for extending BPMN with the knowledge dimension. Still, some of its concepts are changed to accommodate a distinction between knowledge, information and data.

3 Extending BPMN with the Knowledge Dimension

The existence of an organization is achieved by the coordination of member activities and only then by the existence of members themselves, because members without the coordination are individuals rather than an organization [14]. Coordination of BP activities within an organization is achieved with information exchange between organization members that is a basis for knowledge generation and distribution. Knowledge might be considered as one of the BP dimensions, because knowledge is created as a result of process execution, knowledge is used to perform a process and it is distributed among process participants [5]. Extending BP models with the knowledge dimension would provide the following benefits:

- Possibility to identify, plan and manage required knowledge for the role that participates in a particular activity.
- Possibility to evaluate the amount of lost knowledge if a person – owner of knowledge – would leave the organization, in order to identify which tacit knowledge in this case should be transformed into explicit knowledge, such as documents, rules, systems, etc.

- Opportunity to improve understanding about the knowledge usefulness, validity and relevance for particular activities in a process.
- Opportunity to enable competence requirements management and proactive training based on a process reengineering impact analysis.

According to the above-mentioned arguments knowledge and BPs are directly related and their integrated consideration is indispensable. In this paper, we propose a BP modeling technique that supports an integrated consideration of BPs and knowledge. The proposed technique is an extension of BPMN [10, 11 and 19], where the standard notation is supplemented with knowledge modeling related concepts. BPMN was selected as a basis for the notation, because it has become a de facto standard for BP modeling and it is supported by almost all popular process modeling tools. Concepts from KMDL [6] were chosen for the representation of the knowledge dimension, because KMDL supports most of the modeling issues addressed in this paper (Table 1). Nevertheless, the following problems arise when modeling with KMDL: (1) it is not possible to model in an adequate manner decisions, actions and measures, which are causing a sequence of processes; (2) information and data concepts are not distinguished; (3) the method allows modeling knowledge to flow perfectly, but BP modeling with KMDL is challenging - reading and understanding a model requires special thinking and learning. Therefore, in order to make a modeling technique understandable and applicable, we have introduced the main concepts of KMDL (such as an information object, knowledge object, type of knowledge conversion) into BPMN with a few additions and changes in graphical representation. Additions are made because KMDL does not differentiate between data and information. However, data are also considered for knowledge generation, distribution and utilization in the context of collaboration between a person and the computerized part of an IS.

3.1 Data, Information and Knowledge

Scientific literature offers several interpretations of terms like data, information and knowledge [20]. In this section we provide our interpretation of these concepts.

*(I) Data***.** Information theory considers *data* as a functional value of information used for the actions of an interpretation device [14]. D*ata* participate in processes within an object and play the role as a "thing for itself". This is a set of facts which do not have concrete interpretation, such as data in system repositories and data bases or facts inside the human brain. From a communication perspective, a computer based part of information systems interacts with *data* and a person enters *data* into them, because the computer system does not meaningfully interpret data. In contrary, if a person has knowledge that enables him/her to understand and interpret data provided by the computer system, *information* can be received from the system while assigning a meaning for the *data* (Fig.1).

*(II) Information***.** Scientific literature proposes different definitions of information [15, 16, 17, 18 and 20] that are relevant in certain application domains. In the context of human information, proceeding information is an instrument of a knowledge transfer. Information can also be regarded as a nonmaterial entity which allows to describe real (material) and mental (nonmaterial) entities with any degree of precision [14]. Information can be shared in different ways, e.g., via verbal communication, sharing information stored in a data base, externalizing information in paper format, using

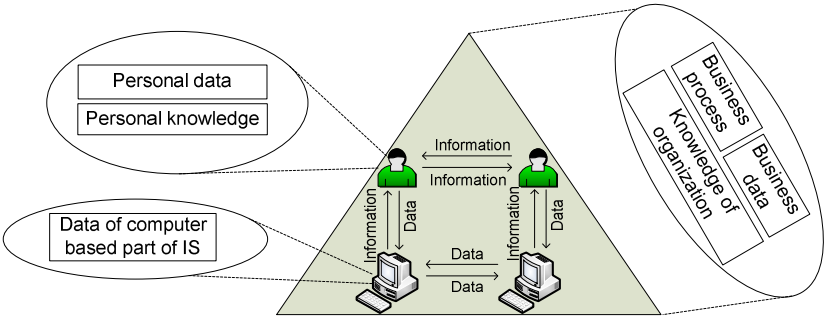


Fig. 1. Data, information and knowledge circulation in organization

information as an electronic document. In the proposed BPMN extension *information* concept is modeled as an information object and it is interpreted as externalized knowledge or meaningful data used for a knowledge transfer between subjects and activities. Non-material information used within communication is not currently presented in the model.

(III) Knowledge. A wide variety of knowledge definitions are used in different domains and environments. The knowledge concept is associated to such terms as skill, capability, competence, and experience. In this paper we consider knowledge as a combination of information, experience and insights a person has obtained from experience or via education and training [4, 16]. Knowledge is owned by a person and it has subjective characteristics. It means that surrounding objects (people, books, information systems, etc.) own only potential knowledge and there is still a question whether the knowledge of one object will become the knowledge of another object. In other words, when a member of an organization is trying to pass his/her knowledge to another member, for the receiver this is just a set of information. The receiver needs to execute a set of internal activities to transform the received information into knowledge. Thus, BP participants interact with information during process execution and everyone can use the received information to create his own knowledge (Fig.1.).

3.2 Knowledge Related Concepts in the Proposed Notation

The proposed technique is built on the basic elements of BPMN [10] that supports representation of all BP related aspects. The knowledge dimension is introduced using the concepts from KMDL with some additions and a new graphical representation. Table 2 compares KMDL and extended BPMN. Every process and activity (subprocess) is defined as either knowledge intensive or not knowledge intensive. The border line between knowledge intensity and non-intensity is fuzzy [12]; we consider an activity as knowledge intensive if it corresponds to any of the knowledge conversion types [20].

Knowledge is associated with a specific person and linked to a role in KMDL. In the extended BPMN knowledge is associated to an activity and additionally it is related with a role (owner of knowledge). A new element *Data* is introduced.

Table 2. Visual representation of elements - differences between KMDL and BPMN extension

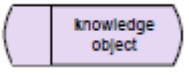
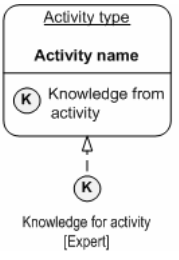
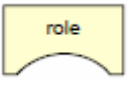
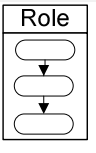
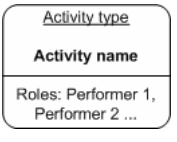
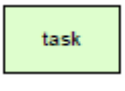
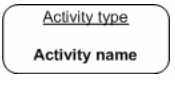
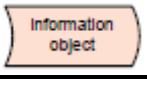
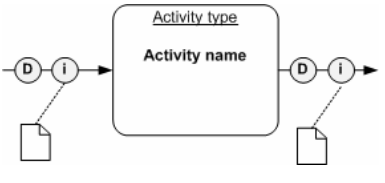
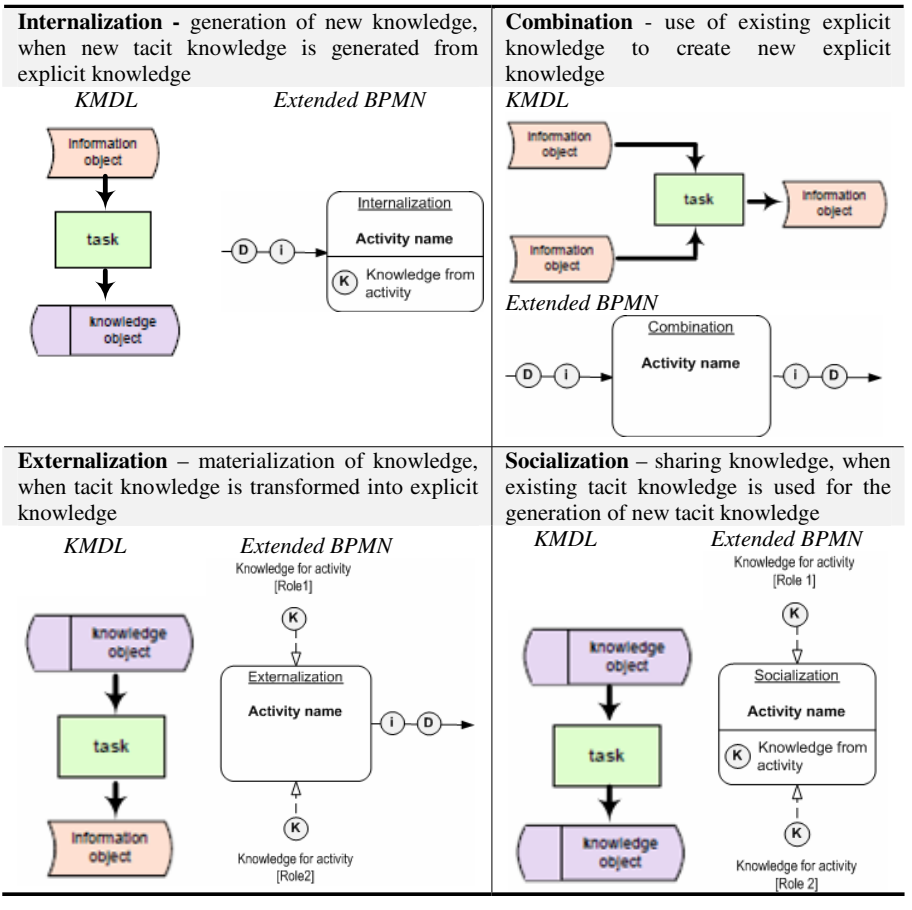
KMDL element	Element in BPMN extension
Knowledge object	
Knowledge object describes tacit knowledge of persons.  Task requirements define tacit knowledge that is necessary for a position working on a concrete task. 	Tacit knowledge required for execution of activity (knowledge for activity) defines activity requirements and is related to an activity. The role that owns - knowledge is shown in brackets. During activity new tacit knowledge can be generated (knowledge from activity), it is situated within activity. It can also form input knowledge for other activities within the process. 
Role	
Roles are taken by persons and have knowledge objects assigned to them. 	All knowledge types are linked to role, because a particular person is not presented in process model. However, the person is fulfilling some role and if the relationship between the person and the role is recorded, it is possible to derive specific knowledge associated to the person. If activity is executed by several roles, then it is presented in the lane without a title and performer roles are included into the activity.  OR 
Activity	
A task is defined as an atomic transfer from input to output, represented as information objects. 	The task equivalent is activity. If activity includes a knowledge intensive process, then the knowledge conversion type is added on top of the element. 
Information object and data object	
Information object is the basis for the creation of new knowledge objects. The creation of new information is done by externalization or combination. It is stored by electronic media or written down in documents. 	Additionally to KMDL definition we distinguish data (D) and information (I) objects according to definitions set in Section 3.1. 

Table 3. Types of knowledge conversions (Knowledge intensive process types)



A data element illustrates potentially meaningful data for new knowledge or information generation during an activity. In KDML different elements are used for the knowledge intensive process type and task (activity). In the extended BPMN, the knowledge conversion type is the attribute of an activity (Table 3).

4 Illustrative Example of the Use of Extended BPMN

To illustrate the knowledge dimension in real-life processes, we refer to a case study where BPMN business process models were created for the Spanish Bioinformatics Company. In section 4.1, we present the process model of a bioinformatics data base integration project (Fig. 2 and Fig.3). This is a knowledge-intensive process. Therefore, a knowledge conversion type is added for each activity. Information or knowledge objects are considered, if they are required for the execution of the activity and are created as a result of the activity. Presented data objects relate to communication between a person and the computer system or the computer system and another computer system.

In section 4.2, we present a model for a sample registration process (Fig.4). This is not a knowledge-intensive process, so BPMN models are extended only with data, information, and knowledge objects that are necessary for the execution of activities and are created as a result of activities. The model is also used for illustrating the use of knowledge dimension in business process reengineering.

4.1 Knowledge Dimension in Knowledge-Intensive Processes

One of the goals in the case study was to investigate how data base integration could help to automate the creation of a gene analysis final report which was a very time consuming task requiring intensive manual work that often caused errors. Before conceptual modeling, there was a need to understand the whole process of the human gene analysis because of the following reasons: (1) not all involved parties knew the whole process of the human gene analysis, so it was difficult to understand terms, situation and requirements; (2) the primary source of data was not confirmed; it was possible that some data were used electronically in the beginning of the process, but later printed out and distributed in a paper format. In this case, data integration should have been done during the first process activity, when data appeared; (3) ad-hoc explanations of processes could lead to misunderstandings or incorrect interpretations; (4) it was difficult for new team members to deeply understand the subject, as they could collect the background knowledge only from books (about genetics' basics) or from other people. BPMN was chosen as the modeling notation because it was simple enough to be understandable for biologists who had no BP modeling experience; and in case of BPMN the IT team members were able to use modeling results as an input to requirements engineering activities. The produced models were also used as an input for conceptual modeling and the training of new employees.

In this paper, we show how the existing BPMN models might be enriched with the knowledge dimension in order to understand the required knowledge and competences

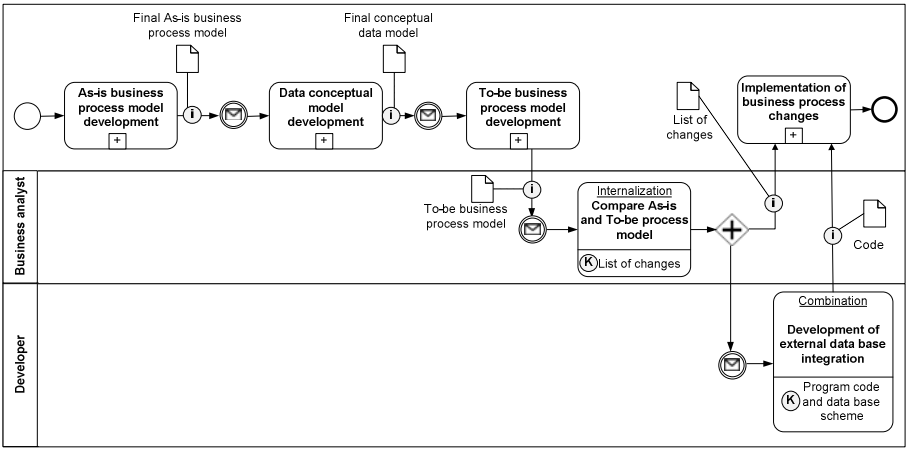


Fig. 2. The development process of the data base integration project at the Bioinformatics Company

in each process step. BPM results in the described project were successful despite the fact that this was the first BPM experience for all team members except the business modeler. All team members got quickly used to basic BPMN concepts and they found models to be understandable and readable. Therefore, we tried to keep models and the notation trivial and close to ones that were created during the project. Figure 2 shows the development process for the described project with added information and the knowledge dimension at the highest level of abstraction.

The first lane is without a title, because it contains activities involving several participants (specified in particular sub-processes) (Fig. 2 and Fig. 3). The development process starts with an As-Is business process model (BP) development activity. The result of this activity is an information item – *As-Is BP model*, which is used as input information for Data Conceptual Modeling. Then data conceptual modeling results in an information item – *Conceptual data model* that is used for the To-Be BP model development. When a BP analyst receives information about the To-Be process model and As-Is process model, he can compare these models and generate knowledge and information about the list of changes. Knowledge about the list of changes is owned by the business analyst and kept in mind, but the information object or materialized knowledge is distributed to other team members, who can internalize this information to generate their own knowledge. Knowing the Conceptual data model and To-Be process model the developer can generate code and data base schema (create new knowledge for himself and an information object for other team members).

In Figure 3, we explore As-Is BP model development, which is a knowledge-intensive process. For example, process modeling starts with Initial investigation – meeting between the University team (Modeler, Biologist) and the Bioinformatics

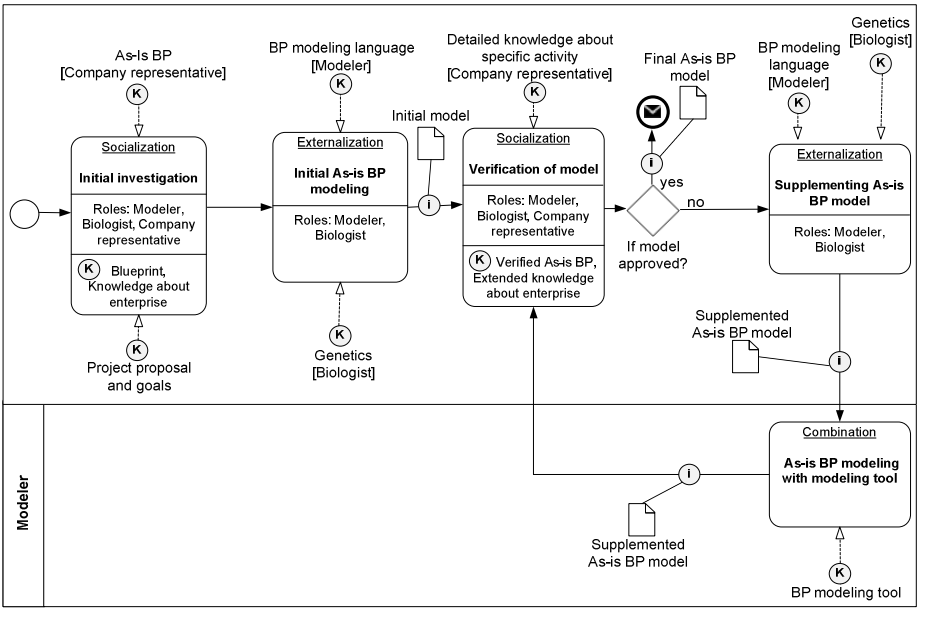


Fig. 3. As-Is BP modeling with knowledge dimension

Company's representatives, where a high level process of human gene analysis is introduced. This is a Socialization type of knowledge conversion, where the Company representative owns knowledge about existing processes and all team members own knowledge about the project proposal and goals. During a meeting this knowledge is presented and discussed with other participants. The result of the meeting is a common understanding about further activities of the analysis (new knowledge for all participants) and general knowledge about the enterprise. Tacit knowledge created as a result of the process is situated within activity element and also forms input knowledge for further activities. The rest of the model is built in a similar manner.

4.2 Knowledge Dimension in Routine BP Modeling

A sample registering process is a sub-process of the gene analysis in the Bioinformatics Company. The goal of this sub-process is to register the received blood or skin sample and prepare the Lab register – a document that describes the sequence of lab activities and responsible persons.

After the data base integration project, the sample registering process might be changed as shown in Figure 4. The Internal Client's data base is integrated with the ERP system, so a receptionist does not need to enter data into the ERP system. It means that the receptionist does not need the knowledge about ERP system. Protocol is selected and the Lab register produced automatically – the employee does not need the knowledge about analysis type-protocol mapping and MS Word.

It is obvious that process changes can remove or add some required knowledge. Knowing a process reengineering impact on the knowledge dimension, the company can change competence requirements for the involved roles or evaluate the relevance of

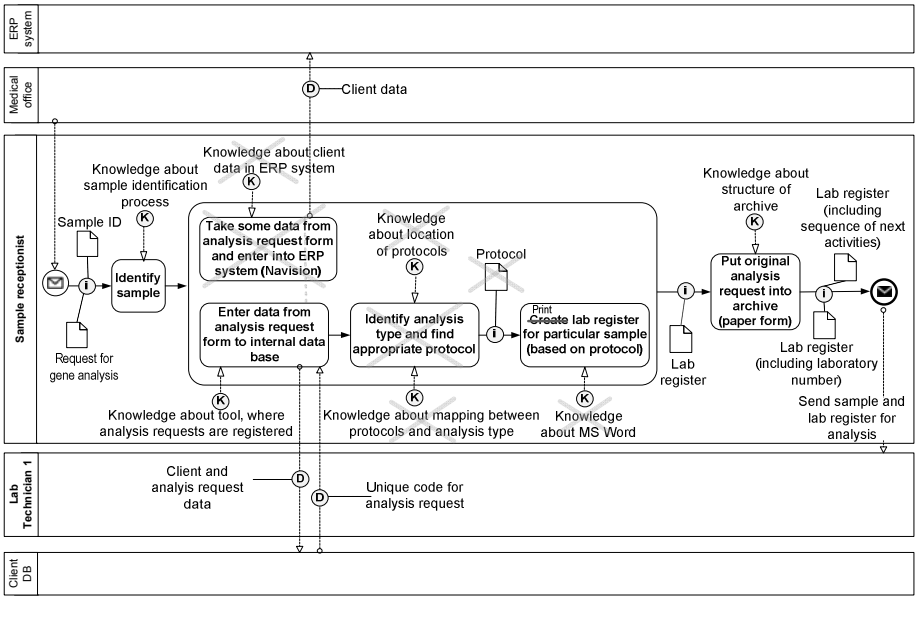


Fig. 4. Sample registration (To-Be) process with changes in process and knowledge

existing competences. It means that increased competence requirements increase the cost of human resources for some roles. In the presented example, competence requirements (required knowledge) are reduced, so the company can hire a cheaper resource.

Using a process model, the company's representatives can also understand that knowledge about the client registration in an internal client's system should be improved after process reengineering and data base integration. It means that the needed training of the impacted personnel can be planned in time during the process modeling phase.

5 Conclusions and Future Work

In the paper we analyzed the existing situation in the area of knowledge oriented BP modeling and process oriented knowledge management. Based on this analysis requirements for knowledge representation in BPM were identified and the necessity of the knowledge (including data and information) dimension justified. We presented BPMN extension with the knowledge dimension which integrates BPMN and KMDL. This approach could be applied for a more effective and efficient BP flow and knowledge modeling, analysis, and planning. The proposed technique is illustrated by a practical example from Bioinformatics area which clearly shows that the technique can be applied in both knowledge-intensive and ordinary processes. The case study also presents the impact of process reengineering on the knowledge dimension in BPMN. The utilization of knowledge dimensions helps to plan the training and changes in required competences and resources already during business process modeling phase.

The proposed approach does not present non-materialized information used for interaction, knowledge flow between particular team members, and it lacks information about the knowledge structure, therefore future research can involve: (1) integration and separation of different knowledge types (like experience, basic knowledge, general knowledge [4]); (2) knowledge repositories and geographical location of knowledge; (3) differentiation of knowledge required for the role and knowledge belonging to a particular person; (4) representing more than two types of conversions taking place in a single activity.

Acknowledgments

Initial BPMN models were created together with Dr. O. Pastor, Dept. of Computation and Information Systems, and consultant - biologist A. M. Levin, both from Valencia University of Technology, Spain, and helpful partners from DNA-sequencing Bioinformatics Company. Valuable comments on the draft of the paper were given by Prof. J. Grabis and L. Penicina, Riga Technical University, Latvia.

References

1. Alavi, M., Leidner, D.E.: Knowledge management systems: issues, challenges, and benefits. Communications of the AIS 1(2) (1999)
2. Weidong, Z., Weihui, D.: Integrated Modeling of Business Processes and Knowledge Flow Based on RAD. In: International Symposium on Knowledge Acquisition and Modeling, Wuhan, China, pp. 49–53 (2008)

3. Zhaoli, Z., Zongkai, Y., Qingtang, L.: Modeling Knowledge Flow using Petri net. In: International Symposium on Knowledge Acquisition and Modeling, Wuhan, China, pp. 142–146 (2008)
4. Hua, J., Zhilou, Y.: Knowledge Management based on Two-dimensional Synthesize Method in enterprise. In: International Conference on Computer Science and Software Engineering, Wuhan, China, pp. 362–367 (2008)
5. Heisig, P.: The GPO-WM® Method for the Integration of Knowledge Management into Business Processes. In: 6th International Conference on Knowledge Management (I-KNOW 2006), Graz, Austria, pp. 331–337 (2006)
6. Gronau, N., Korf, R., Müller, C.: KMDL-Capturing, Analysing and Improving Knowledge-Intensive Business Processes. Journal of Computer Science 4, 452–472 (2005)
7. Woitsch, R., Karagiannis, D.: Process Oriented Knowledge Management: A Service Based Approach. Journal of Universal Computer Science 11(4), 565–588 (2005)
8. Scholl, W., König, C., Meyer, B., Heisig, P.: The Future of Knowledge Management. An international Delphi Study. Journal of Knowledge Management 8(2), 19–35 (2004)
9. Gronau, N., Weber, E.: Management of Knowledge intensive Business Processes. In: Desel, J., Pernici, B., Weske, M. (eds.) BPM 2004. LNCS, vol. 3080, pp. 163–178. Springer, Heidelberg (2004)
10. Business Process Model and Notation (BPMN), FTF Beta 1 for Version 2.0 (2009), http://www.omg.org/spec/BPMN/2.0
11. Owen, M., Raj, J.: BPMN and business process management; Introduction to the new business process modeling Standard (2003), http://www.bpmn.org/Documents/6AD5D16960.BPMN_and_BPM.pdf
12. Donina, D.: Modelling knowledge intensive processes in organizations, Master thesis, Riga Technical university (2008)
13. Scholl, W., König, C., Meyer, B., Heisig, P.: The Future of Knowledge Management. An international Delphi Study. Journal of Knowledge Management 8(2), 19–35 (2004)
14. Янковски, С.: Концепции общей теории информации, http://n-t.ru/tp/ng/oti.htm
15. Moon, T.: course, Information Theory, Electrical and Computer Engineering, Utah State University, http://ocw.usu.edu/Electrical_and_Computer_Engineering/Information_Theory/?searchterm=None
16. Making Sense of Data and Information, produced by Elearn Training Company and published by Elsevier, http://searchdatamanagement.techtarget.com/generic/0,295582,sid91_gci1283796,00.htm
17. Carter, T.: An introduction to information theory and entropy, http://astarte.csustan.edu/~tom/SFI-CSSS/info-theory/info-lec.pdf
18. Peter, A.C.: Control Information Theory: The 'Missing Link' in the Science of Cybernetics. System Research and Behavioral Science 24(3), 297–311 (2007)
19. Business Process Modeling Notation Tutorial, http://www.bpmn.org/Documents/FAQ.htm
20. François, C.: International Encyclopedia of Systems and Cybernetics, 2nd edn. K.G. Saur, Munich (2004)
21. Nonaka, I., Takeuchi, H.: The Knowledge-Creating Company. How Japanese Companies Create the Dynamics of Innovation. Oxford University Press, New York (1995)

Perspectives for Integrating Knowledge and Business Processes through Collaboration

I.T. Hawryszkiewycz

School of Systems, Management and Leadership
University of Technology, Sydney
igorh@it.uts.edu.au

Abstract. Collaboration is now playing a greater role in business processes, where knowledge workers leverage knowledge to develop innovative products and services. Such business processes go beyond the goal of simply achieving a well defined outcome at minimum cost using well defined tasks. They place more emphasis on collaboration and knowledge sharing and ways to change processes as collaboration evolves. The paper calls for greater emphasis on perspectives other than process flow in process design. These are knowledge, social structure, business activity, organization and technology. The paper describes process design ways to combine these perspectives into a holistic model and converting the model a collaborative infrastructure that allows users to align collaborative technologies to their collaboration.

Keywords: Collaboration, Knowledge, business process, perspectives.

1 Introduction

Broad business strategies influence strategies followed in information system development. One common business strategy is to obtain competitive advantage by creating efficiencies through process automation. This strategy focuses on the process workflow perspective by automating the sequencing of process tasks. An increasingly evolving strategy is towards encouraging the collaboration that leads to obtaining competitive advantage through innovation. As a result processes are becoming more dynamic and emergent [1] to respond to any opportunities that arise through collaboration.

Such dynamic processes require different supporting technologies from those needed to support process flows. To design such processes and their supporting technologies requires design emphasis on additional perspectives than simply transaction efficiency. It requires greater emphasis on perspectives for leveraging knowledge into business processes. It is applicable to those processes that support the knowledge sharing and collaboration necessary for innovation. The importance of collaboration and social networking is further substantiated by Pralahad and Krishnan [2] who see an increasingly important role for collaboration in business networking. This is seen as part of a broader vision of business evolution known as Enterprise 2.0 [3]. The Enterprise 2.0 vision describes in relatively abstract terms what new businesses will look like. It sees collaboration growing between organizational units within and between organizations. At the same time collaboration changes as new opportunities

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 82–93, 2010.

arise. Enterprise 2.0 suggests that competitive advantage can be obtained by using new technologies such as those currently emerging through Web 2.0 to both support collaboration and its changing nature.

In summary, process design in this paper focuses on knowledge workers [4]. These workers must quickly assess complex business situations and respond to them. Efforts to reengineer the work of knowledge workers into prescribed forms have proven unworkable [4]. Studies have shown that knowledge workers are characterized by greater emphasis on continuously changing social connectivity and interactivity.

Rinkus [5] and others, for example, see social issues as primary in health systems and have developed the HCDID methodology that has user communication as a primary perspective in design. Hence perspectives that focus more on knowledge sharing and social structure are becoming more important in process design. Semantics of change can then be defined as driven by perspectives other than process efficiency. Change can be driven from the social perspective as for example creating a new team to make a proposal. The impact of this change on other perspectives can then be evaluated as for example providing the team with needed knowledge. Alternative an emerging knowledge requirement may be converted to its impact on process and social structure, as for example what expertise is needed to create the knowledge.

The paper first proposes a choice of perspectives. It shows how the perspectives can be integrated into a methodology and defines a design process based on the perspectives. The perspectives proposed are knowledge, social structure, organizational structure, business activity and process sequence. Increasing emphasis on achieving competitive advantage is resulting in more attention to create innovative organizational structures by facilitating collaboration and knowledge sharing. The goal is to develop an infrastructure that provides the commands for knowledge workers to align collaborative technology to changing collaborative processes.

2 Choice of Perspectives

The choice of perspectives is governed by the emerging enterprise trends. These include:

- The emergence of process ecosystems [6], where links between the different processes are continually changing and awareness must be maintained between process participants to keep track of outcomes in distant units that may impact on their own work,
- The trend to a more service oriented environment where systems must continually respond to changing customer needs requiring the continuous sharing of knowledge across units through collaboration and socialization in the business processes,
- Greater client involvement in product design [7] where solutions are created through collaboration between supplier network and the customer network. Often there is a major supplier who originated a project and who then builds and coordinates a network of providers and customers to develop solutions that can provide continually evolving services and co-created services to customer
- Greater emphasis on getting expert advice through business networking,

- Providing the ability to learn to achieve social capital [8] that is considered as essential in innovative learning organizations, and
- Greater collaboration and emphasis on authentic team work [9] where team members collaborate by sharing their expertise towards common goals to create jointly owned artefact.

These trends result in greater emphasis on collaboration and knowledge sharing within the business process. Hence it is crucial to include social networking as a significant part if process designs are to cater for knowledge workers, who as a rule do not follow prescribed processes. On the other hand, knowledge workers require support to enable them to quickly change their social work connections to meet new and often unanticipated business opportunities and quickly adapt to changing situations. They should be able to do so in a way that they can quickly comprehend how to adopt any new technology, and assimilate it in their work.

Knowledge is a primary perspective as it often determines innovation capabilities. Hence social structures and leadership become important in business processes to nurture knowledge sharing through team structures. In summary, the perspectives proposed here are:

- The **business activities** and their actions and outcomes,
- The **process workflow** or sequence of activities and the interdependence between activities,
- The **social structure** that describes roles and their responsibilities and the assignment of roles to individuals and the relationships between them. This is critical in the design of the collaborative infrastructure as it defines the specific collaboration
- The **knowledge** created and used during the activities,
- The **organizational perspective** in the kinds of teams support or leadership provided to support collaboration and innovation,
- The **technology** to support the collaboration, which is needed to share and create knowledge.

The paper considers how to integrate the perspectives by defining semantics that are meaningful to process participants. These both introduce a language and structure that enables meaningful communication that integrates the concepts into a holistic system.

2.1 Integrating the Perspectives

The objective is to provide integrate the perspectives into a holistic model. This requires:

- The development of the criteria to be used in the integration,
- The creation of an architecture that is implementable, and
- Defining the concepts for each perspective and their integration.

Figure 1 defines the main perspectives and criteria for linking them. The criteria are shown on the links between the perspectives. For example, business activities are related to the social structure through responsibilities allocated to roles in the activity.

They are related to knowledge as they both need knowledge and use it to create new knowledge. Social structures are related to knowledge as knowledge is created through social interactions in the business activities supported by technology. The business process organizes the activities into a process.

The next is to define an architecture that combines these dimensions. This is called a blueprint based on design science ideas.

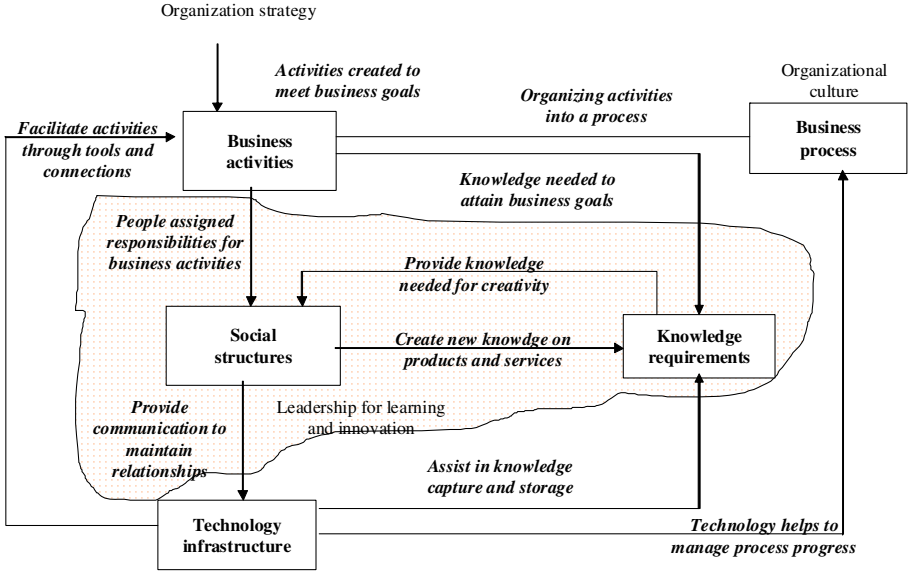


Fig. 1. Criteria Combining the Perspectives

3 An Architectural Blueprint for Design

Collaboration support must go beyond many current informal approaches of encouraging meetings and e-mail communication. These often focus on making collaborative technologies available but leaving it up to users to identify where to use the technologies, without identifying any particular long-term business benefit. Such uncoordinated use can be counterproductive as suggested by Hansen [12] as people may spend considerable time communicating without attaining a useful outcome.

The design blueprint proposed here follows the suggestion by Pisano and Verganti [11], who define a number of different strategies for collaboration support. They propose the idea of creating a *collaborative architecture* that customizes collaboration for a given network. The business value of the collaborative architecture is clearly identified by the kind of knowledge created during the collaboration.

The paper addresses ways to create such collaborative infrastructures focusing on the evolving service environment [7]. The paper distinguishes between business architecture, collaborative architecture and collaboration infrastructure. In this terminology:

- Business architecture are the way work is organized within business activities to achieve a particular objective. This can focus on delivering new products or services or in the collaboration needed to form business networks to deliver innovative services,
- The collaborative architecture is the ways that business entities collaborate within the business architecture to achieve their business goals, and
- The collaboration infrastructure that supports the collaboration including technology support through social or other software.

4 Design Process

The method proposed here focuses on defining the requirements for collaborative business activities, including their collaborative architecture. The paper describes a method that follows this structure to support business collaboration focusing on ways to support service industries. In selecting such a method a criterion is to decide on a driving perspective. It differs from many of the current design methodologies as it places greater emphasis on social structures early in the design. A further difference is that the emergent nature of collaborative processes is not easily modelled using the more structured modelling tools found in most methodologies. The collaborative nature of the target systems requires greater emphasis on knowledge sharing and the way collaborators can use and create knowledge in their business.

Figure 2 illustrates a simple form design process that commences with business activities and knowledge. More details can be found in [12]. The design steps in the design process include all the perspectives. The design process is not necessarily sequential although the number in each step indicates a possible sequence. Each design step uses some guidelines and produces an outcome.

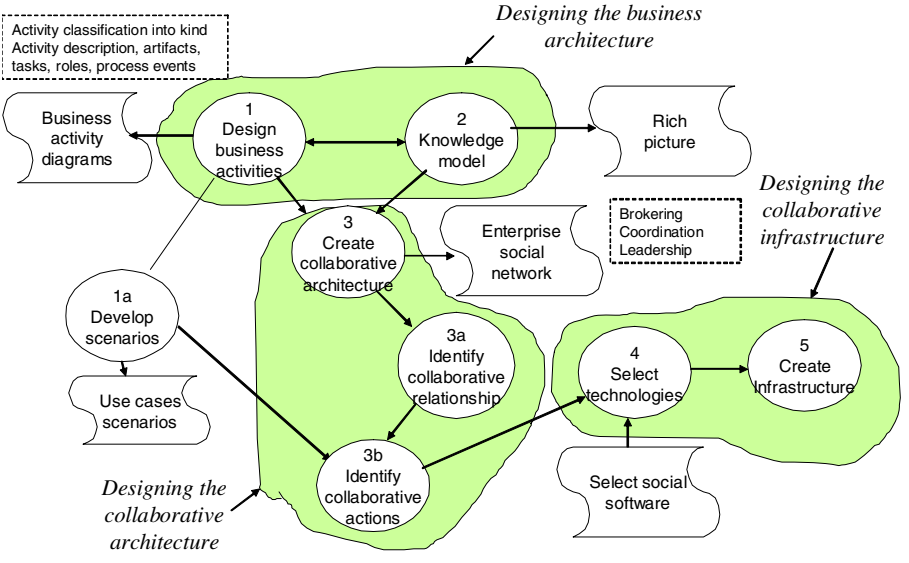


Fig. 2. Design process

The process begins with defining the business activities (step 1 i9n Figure 2) and their knowledge requirements (step 2). Scenarios (step 1a) can be developed as part of the business activities. Next we the collaborative architecture is defined (step 3). The collaborative architecture defines the roles, their responsibilities and the relationships between them.

The next sections describe the modelling concepts for the various concepts and their integration.

4.1 Modeling the Business Activity Perspective

The concepts for business activities can be used to define business activity diagrams. This can both serve as a specification or as a cognitive view of the business system. Figure 3 illustrates a business activity diagram. Here the clouded shapes represent activities, black dots represent roles, and disk shapes represent artefacts. Figure 3 is a model of a typical process of an organization responding to a tender. The main activities are:

- Developing a technical solution, which may be a building a road design,
- Developing the cost response part,
- Development of an implementation plan including fitting in with local factors such as construction rules and environmental standards.
- Assembly of the three parts and their combination with the personnel records of people who will be involved in the response, and the track of the responding organization.

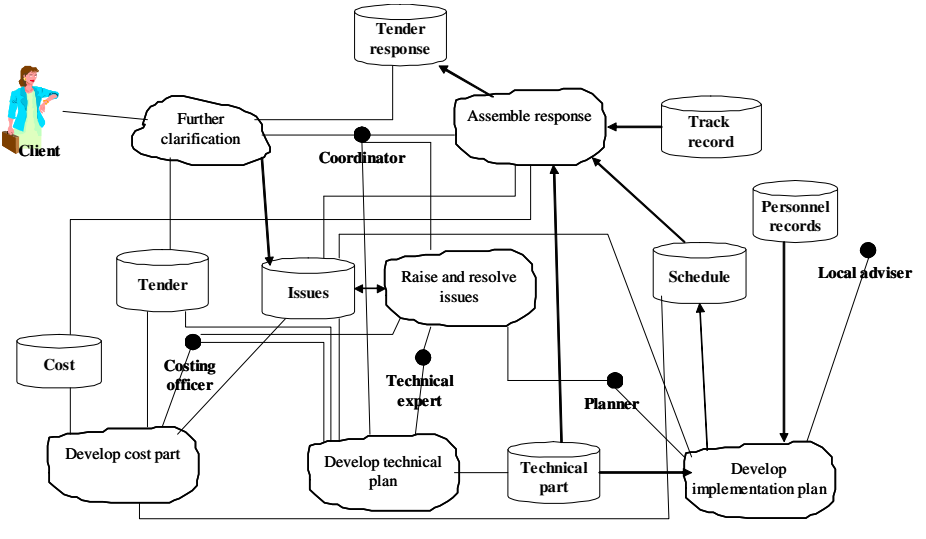


Fig. 3. Business activities in tender response

There are also formal collaborative activities such as resolving issues and ensuring consistency between the response parts. The degree of cooperation here is very high. Often there is need to get hold of experts in any of the three parts and to get clarifications from the client. The processes and activities can quickly change. There may suddenly be a need to make an environmental study or seek special approvals from local authorities.

Figure 3 shows part of the knowledge and social perspectives. It shows the artifacts or explicit knowledge that is available. This is the tender, various personnel records, track record of previous projects and so on. It also shows the roles needed in each activity to organize the activity and produce the necessary outputs.

4.2 The Knowledge Perspective

The knowledge perspective is generally not highly structured and rich pictures are proposed as the modeling tool to emphasize the tacit part of knowledge. Figure 4 gives an idea of the kinds of knowledge created and used by the different roles in the business activities. The kinds of knowledge created are possibilities through matching different kinds of existing knowledge through socialization and then externalizing it in codesign to create new knowledge, in terms of combining identified services. It shows in an informal manner the kind of interactions and the emergent activities that are often part of knowledge based enterprises. Thus for example the rich picture shows that ideas come up, they can result of suggestions for new initiates that are often further elaborated in meetings.

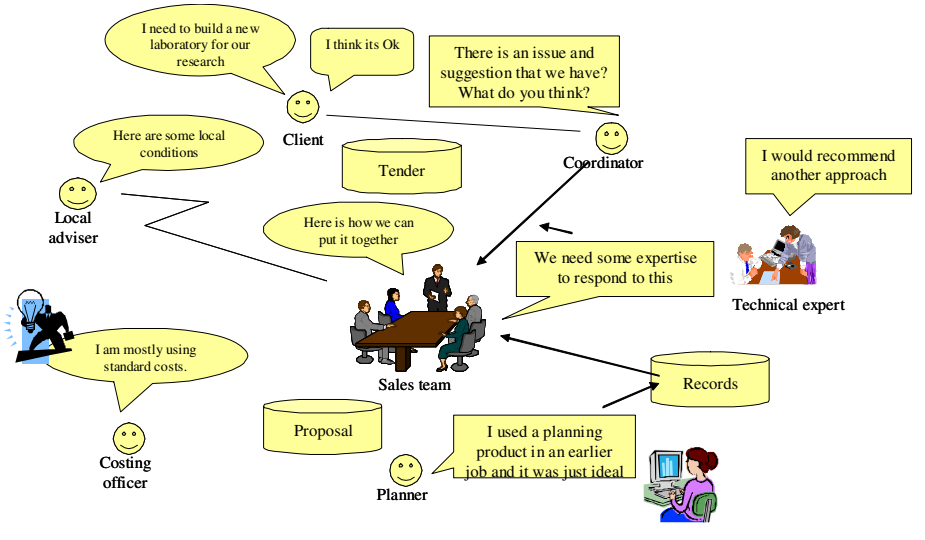


Fig. 4. A rich picture showing knowledge requirements

4.3 Concepts for the Social Perspective

Social network diagrams have been widely used to model relationships between people. These have been extended in a variety of ways to suit different purposes. Business

collaboration requires a clearer definition of what the people do and how they should collaborate in their work. At the same time, the chosen structures must naturally support the social acceptance of any new design. The extension is relatively simple. The link to business activities is through roles. The roles have defined responsibilities and the communication that forms part of these responsibilities is also included in the ESN.

Figure 5 shows the notation used in ESNs. Roles are shown by the black dots whereas individuals are shown as a face. The labels attached to each role show the role responsibilities and the labels attached to the lines joining the roles show the interactions between the roles. It is these interactions that often capture much of the knowledge needed in future decisions. Thus in Figure 5, X1 and X2 occupy roles A and B respectively and through their interaction create some knowledge. Often this created knowledge is based on interpretations by the individuals using their tacit knowledge. In implementation such created knowledge can be captured using the different Web 2.0 systems now becoming more commonly available. Dotted lines are sometimes used to show the informal relationships in the system.

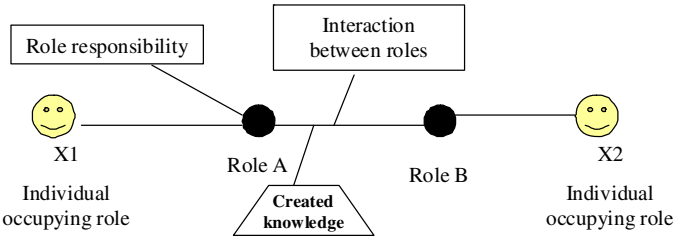


Fig. 5. ESN Notation

The ESN for tender evaluation is given in Figure 6. It shows the roles and their interactions and potential knowledge created during these interactions.

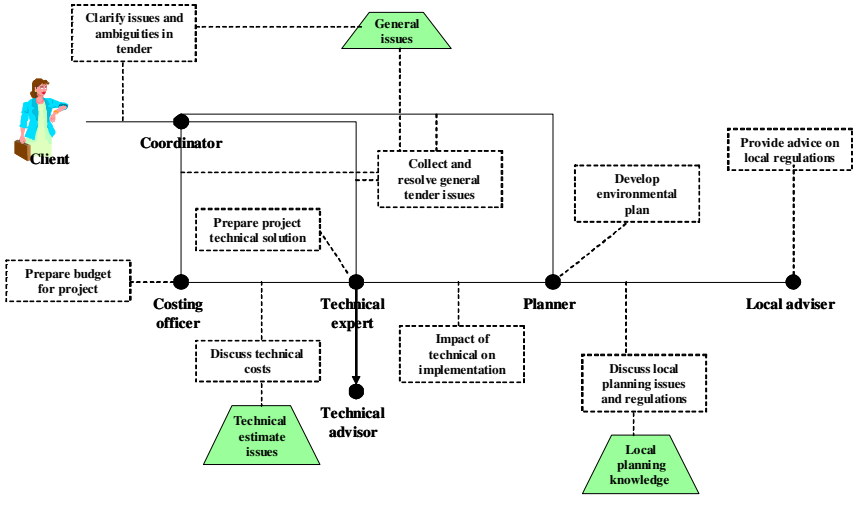


Fig. 6. ESN for tender evaluation

Completeness can be assured through checklists that ensure that all knowledge elements and roles are included in the ESN.

5 Conversion to the Collaborative Infrastructure

Technology requirements here are classified into the infrastructure and the interfaces provided to users. One requirement is a social database that stores the relationships illustrated in Figure 7. The requirements are:

- Ways to support the social network and capture the knowledge,
- Business activities and roles created in these activities
- Governance of assigning people to roles.

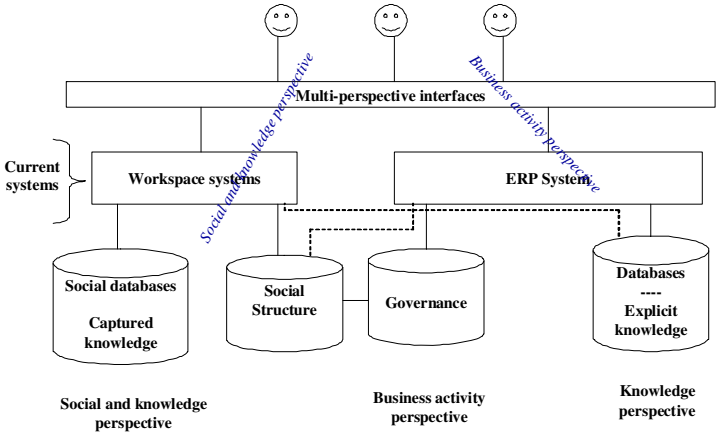


Fig. 7. Current technologies

A broad analysis of current technologies is illustrated Figure 7. There is usually a division between that part of a technology infrastructure that supports business activities and that which supports social structures with knowledge often shared between the two. Business activities are usually supported by ERP systems together with databases that hold the explicit knowledge. Social activities are usually supported by workspace systems of Web portals and often contain knowledge captured as part of social interactions. The challenge is to somehow combine the two parts into an integrated structure.

5.1 Combining the Perspectives to Create an Infrastructure That Integrates Knowledge, Social Structure and Activity

In this sense we look at each role from the knowledge perspective and decide on the support needed by the roles to support its activities. An enterprise social network (ESN) diagram is used for this purpose. The architecture here is to create the collaborative

infrastructure composed of the kind of services found in Web 2.0 or software services. The idea is shown in Figure 8.

Examples here include:

- A wiki is setup to construct the local environmental plan, with experts and internal staff contributing to it,
- A general issues discussion that involves most of the task leaders.

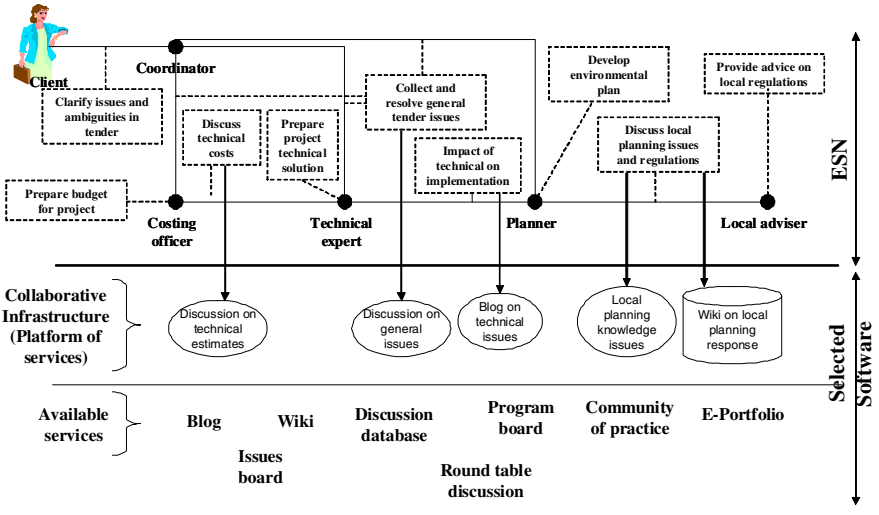


Fig. 8. Enterprise Social Network

6 Infrastructures for Supporting the Collaborative Architecture

Lightweight platforms that support adaptive systems and support user activated process realignment should include the concepts defined for the collaborative model while providing commands to easily create and change the structures of workspaces.

Commercial systems in this area focus on middleware software that provides the commands that allows users to use the middleware functionality to create workspaces. Furthermore, it should allow users to change the workspaces as work practices change. Many manufacturers are now providing ways to integrate the kind of software with enterprise applications. A typical example here is Websphere provided by IBM. The challenge in many such systems is to provide ways to share knowledge across activities. They provide access to corporate databases but often do not support the sharing of knowledge collected in the course of knowledge work in identifying and solving problems, and making decisions.

6.1 Groupware Platforms

One question is what kind of commands should be provided to knowledge workers to realign collaborative support to their changing collaborative activities. Our experimental

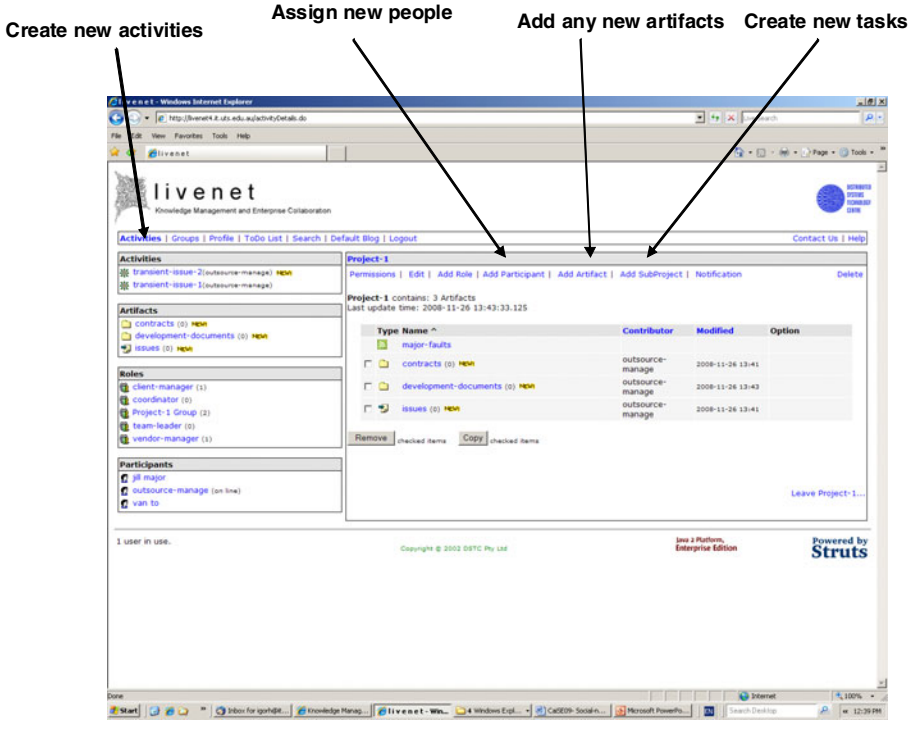


Fig. 9. A Demonstration Workspace

system, LiveNet, demonstrates the kind of support needed by workspace systems. Figure 9 shows the LiveNet interface and its typical commands.

It provides a menu that can be used to create new collaborative objects, including activities, roles, and artefacts. It also enables people to be assigned to the roles. Apart from these elementary operations the system includes ways to implement governance features as for example allowing roles limited abilities to documents. The system includes support for sharing artifacts across workspaces and a permissions structure to control such sharing. Social software such as blogs or discussion systems is supported and can be shared across workspaces.

It provides a menu that can be used to create new collaborative objects, including activities, roles, and artefacts. It also enables people to be assigned to the roles. Apart from these elementary operations the system includes ways to implement governance features as for example allowing roles limited abilities to documents.

Commercial systems in this area focus on middleware software that provides the commands that allows users to use the middleware functionality to create workspaces. Furthermore, it should allow users to change the workspaces as work practices change. Many manufacturers are now providing ways to integrate the kind of software with enterprise applications. A typical example here is Websphere provided by IBM.

7 Summary

This paper focused on identifying perspectives for designing processes that focus on innovation within complex business environments. It introduced the idea of integrating social networking into the process and suggested collaborative infrastructure as a way to realize such processes. It then described the important factors in combing perspectives into a methodology. These included identifying the criteria for the links, links between the concepts in each perspective and an implementable architecture.

The next step is to add organizational structure and human relations factors as perspectives. This will extend the model to include organizational unit responsibilities and the governance structure. It will also enable the model to clearly show team structures and their organizational representations as well as the extent of knowledge sharing across the whole organization. The human relations factors will provide ways to match people to role in business activities.

References

1. Hall, J.M., Johnson, M.E.: When Should a Process be Art. Harvard Business Review 84(2), 58–65 (2009)
2. Prahalad, C.K., Krishnan, M.S.: The New Age of Innovation. McGraw-Hill, New York (2008)
3. McAfee, A.P.: Enterprise 2.0: The Dawn of Emergent Collaboration. MIT Sloan Management Review 47(3), 21–28 (2006)
4. Davenport, T.: Thinking for a Living. Harvard Business Press (2005)
5. Rinkus, S., Walji, M., Johnson-Throop, K.A., Malin, J.T., Turley, J.P., Smith, J.W., Zhang, J.: Human-Centered design of distributed knowledge management system. Journal of Biomedical Informatics 38, 4–17 (2005)
6. Vidgen, R., Wang, X.: From business process management to business process ecosystem. Journal of Information Technology 21, 262–271 (2006)
7. Cova, B., Salle, R.: Marketing solutions in accordance with S-D logic: Co-creating value with customer network actors. In: Industrial Marketing Management, vol. 37, pp. 270–277. Elsevier Press, Amsterdam (2008)
8. Hannah, S.T., Lester, P.B.: A multilevel approach to building and leading learning organizations. The Leadership Quarterly 20, 34–48 (2009)
9. Lick, D.W.: A new perspective on organizational learning: Creating learning teams. Evaluation and Program Planning 29, 88–96 (2006)
10. Hansen, M.T.: When Internal Collaboration is Bad for Your Company. Harvard Business Review 84(3), 83–88 (2009)
11. Pisano, G.P., Verganti, R.: What Kind of Collaboration is Right for You. Harvard Business Review 83(8), 80–86 (2008)
12. Hawryszkiewycz, I.T.: Knowledge Management: Organizing the Knowledge Based Enterprise. Palgrave-Macmillan, Basingstoke (2010)

Workflow Time Patterns for Process-Aware Information Systems

Andreas Lanz[1], Barbara Weber[2], and Manfred Reichert[1]

[1] Institute of Databases and Information Systems, Ulm University, Germany
{Andreas.Lanz,Manfred.Reichert}@uni-ulm.de
[2] Quality Engineering Research Group, University of Innsbruck, Austria
Barbara.Weber@uibk.ac.at

Abstract. Formal specification and operational support of time constraints constitute fundamental challenges for any process-aware information system. Although temporal constraints play an important role in the context of long-running business processes, time support is limited in existing process management systems. By contrast, different kinds of planning tools (e.g., calendar systems, project management tools) provide more sophisticated facilities for handling task-related time constraints, but lack operational support for business processes. This paper presents a set of time patterns to foster systematic design and comparison of the different technologies in respect to the time perspective. These time patterns are all based on empirical evidence from several large case studies. Their widespread use will contribute to further maturation of process-aware information systems and related evaluation schemes.

1 Introduction

Formal specification and operational support of the time perspective constitute fundamental challenges for any enterprise information system. Although temporal constraints play an important role in the context of long-running business processes (e.g., patient treatment, automotive engineering, flight planning) [1,2,3], support of the time perspective is rather limited in existing process management systems [1,4] (as opposed to other process perspectives like control flow and data flow). By contrast, different kinds of planning tools (e.g., calendar systems and project management tools) provide more sophisticated facilities for handling time constraints (e.g., periodic activities), but miss an operational support for business processes. So far, there is a lack of methods for systematically assessing and comparing the time capabilities provided by these different process support technologies (denoted as *Process-Aware Information Systems* (PAIS)).

To make PAIS better comparable and to facilitate selection of appropriate PAIS-enabling technologies, workflow patterns have been introduced [5,6,7]. Respective patterns provide means for analyzing the expressiveness of process modeling approaches in respect to different process perspectives. For example, proposed workflows patterns cover control flow [5], data flow [6], exceptions [8], and process change [7]. However, a framework for systematically evaluating PAIS in

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 94–107, 2010.

respect to their ability to deal with the time perspective is missing and is picked up by this paper. Our contribution is as follows: We suggest *time patterns* to foster comparison of existing PAIS with respect to their ability to cover the time perspective of processes. The proposed time patterns complement existing workflow patterns and have been systematically identified by analyzing a large collection of processes in healthcare, automotive engineering, aviation industry, and other domains. The presented work will not only facilitate PAIS comparison in respect to the support of time constraints, but also foster selection of appropriate time components when designing PAIS.

Section 2 summarizes basic notions. Section 3 presents the research method employed for identifying the time patterns. Section 4 describes the proposed time patterns sub-dividing them into 4 categories. We present related work in Section 5 and conclude with a summary and outlook in Section 6.

2 Basic Notions

This section describes basic concepts and notions used in this paper: A *process management system* is a specific type of information system which provides process support functions and separates process logic from application code. For each business process to be supported, a *process type* represented by a *process schema* has to be defined (cf. Fig. 1). In the following, a process schema corresponds to a directed graph, which comprises a set of *nodes* – representing *activities* and *control connectors* (e.g., XOR-Splits or AND-Joins) – and a set of *control edges* between them. The latter specify precedence relations. We further use the notion of *activity set* to refer to a subset of the activities of a process schema. Its elements are not required to be part of a sequence block, but may also belong, for example, to different parallel branches. During run-time *process instances* are created and executed according to a predefined process schema S. *Activity instances*, in turn, represent executions of single process steps of a particular process instance. Activities which shall be executed more than once (concurrently or sequentially) are referred to as *multi-instance activities*.

The patterns introduced in the following can be applied to these granularities, i.e., process schema, activity, activity set, activity instance, and process instance. We use the term *process element* as umbrella for all these concepts.

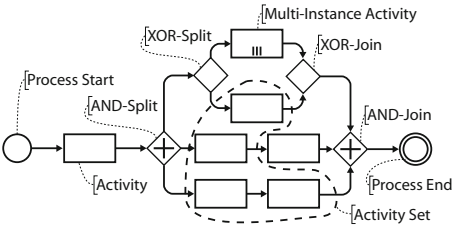


Fig. 1. Core Concepts of a Process Model

3 Research Method

The overall goal of this paper is to complement existing workflow patterns with a set of time patterns suitable to assess how effectively PAIS can deal with time. As motivated in the introduction, adequate modeling and management of temporal constrains will be a key feature of future PAIS, particularly regarding the support of long-running processes involving humans (e.g., patient treatment).

We describe the selection criteria for our time patterns, the data sources they are based on, and the procedure we have applied for pattern identification.

Selection Criteria. We consider patterns covering temporal aspects relevant for the modeling and control of processes and activities respectively. Our focus is on high coverage of real-world scenarios, and not on specific time features of a PAIS like verification of time constraints [4,1,2], escalation management [9], or scheduling support [10,11].

Sources of Data and Data Collection. As sources for our patterns we consider results of case studies we performed in different domains.

One of our data sources is a large *healthcare* project in which we designed core processes of a Women's Hospital [12]. Selected processes were implemented using existing workflow technology. As part of this project time aspects were elicited and documented. In total we consider 98 process models covering both administrative processes (e.g., order handling) and treatment processes (e.g., chemotherapies and ovarian cancer surgery).

As second data source we use process models from *automotive industry*. We consider a case study on electronic change management (ECM) [13]. Corresponding models have been published by the German Association of the Automotive Industry (VDA) [13]. In total this project provides 59 process models.

As third data source serves a case study we conducted with an *on-demand air service*. As part of this project we analyzed flight planning and handling post flight phases. As aviation industry is highly regulated, compliance with standards and regulations, in addition to company policies, is essential (e.g., minimum standards for flight time limitations, or rest time regulations). Many of these regulations contain time constraints to be obeyed.

Our fourth data source are *healthcare processes* from a large Medical University Hospital. It comprises 60 different processes, related to diagnostic and therapeutic procedures in the field of internal medicine (e.g., examinations in medical units like radiology, gastroenterology, and clinical chemistry). Finally, we have deep insight into patient scheduling systems.

Pattern Identification Procedure. To ground our patterns on a solid basis we first create a list of candidate patterns. For this purpose we conducted a detailed literature review and rely on our experience with PAIS. Next we thoroughly analyzed the above mentioned material to find empirical evidence for our time patterns and - if necessary - extend the pattern candidate list. As a pattern is defined as reusable solution to a commonly occurring problem we require each of our time patterns to be observed at least three times in different models of our

samples. Therefore, only those patterns, for which enough empirical evidence exists, are included in the final list of patterns.

4 Time Patterns

As result of our analysis we have identified 10 different patterns which we divide into 4 distinct categories (cf. Fig. 2a). These time patterns constitute solutions for realizing commonly occurring time aspects in PAIS. Pattern Category I (*Durations and Time Lags*) provides support for expressing durations of process elements (e.g., activities) as well as time lags between events (e.g. milestones) or activities. Pattern Category II (*Restrictions of Process Execution Points*) allows specifying constraints regarding possible execution points of process elements (e.g., activity deadline). Category III (*Variability*) provides support for time based variability (e.g., control-flow varies depending on time context). Finally, Category IV (*Recurrent Process Elements*) comprises patterns for supporting recurrent process elements (e.g., periodicity and cyclic flows). Due to lack of space only 7 out of 10 patterns will be described in detail. For the remaining patterns we refer to our technical report [14].

Pattern Catalogue
Category I: Durations and Time Lags
TP1: Time Lags between Activities
TP2: Durations
TP3: Time Lags between Events
Category II: Restrictions of Process Execution Points
TP4: Fixed Date Elements
TP5: Schedule Restricted Elements
TP6: Time Based Restrictions
TP7: Validity Period
Category III: Variability
TP8: Time Dependent Variability
Category IV: Recurrent Process Elements
TP9: Cyclic Elements
TP10: Periodicity

Fig. 2a. Pattern Catalogue

General Design Choices
A.) Parameters of a pattern may be set at different time points a.) At build-time (i.e., during process modeling) b.) At instantiation time (i.e., when a process instance is instantiated) c.) At run-time (i.e., during process execution)
B.) Time parameters can be specified in different time granularities a.) Basic (i.e., years, months, weeks, days, hours, minutes, seconds) b.) System-defined (e.g., business days) c.) User-defined (e.g., Wednesday afternoon)
C.) Patterns can be applied to different process elements a.) Single activity (including multi-instance activities) b.) Activity set c.) Process model d.) Set of process instances

Fig. 2b. General Design Choices

Fig. 2a gives an overview of the time patterns. For each discussed pattern we provide a name, synonyms, a brief description of the addressed problem, design choices, remarks regarding its implementation, examples from our case studies, a reference to related patterns, and known uses of the pattern summarized in a table (cf. Fig. 4 - Fig. 12).

In particular, *design choices* allow for parameterizing time patterns keeping the number of distinct patterns manageable. Design choices not only relevant for a particular pattern, but for several ones, are described only once. Typically, existing PAIS do not support all design choices regarding a specific pattern. We denote the combination of design choices supported by a particular approach as *pattern variant*.

Fig. 2b describes three design choices concerning the point in time when temporal constraints are set, the time granularities supported and the process elements to which the respective pattern can be applied. These design choices are valid for several or all of the patterns, and can be used for parameterizing them. If not all options of a design choice are valid for a time pattern this is described with the respective pattern. Additional design choices, only relevant for a specific pattern or pattern category, are provided with the respective description.

To formalize operational semantics of the time patterns we first introduce the notion of *temporal execution trace* (*trace* for short): We assume that all events related to the execution of a process instance[1] are recorded in a trace together with a timestamp designating their time of occurrence. Informally, temporal execution traces are defined as follows (for a formal definition see [14]):

Definition 1 (Temporal Execution Trace)

1. *An event occurrence is a tuple* $\varphi = (e, t)$ *consisting of event* e *and timestamp* t*, where* t *defines the exact point in time at which event* e *occurred.*
2. *A temporal execution trace* $\tau_S =< \varphi_1, \ldots, \varphi_n >$ *is an ordered set of event occurrences* φ_i*, where the order of* φ_i *in* τ_S *reflects the temporal order in which the events occurred during process execution*[2]*, i.e.*
$$\varphi_k, \varphi_j \text{ with } k < j \Rightarrow t_k < t_j.$$
3. $occurrences(S, e, \tau_S) = \{\varphi \in \tau_S | \varphi = (e, \cdot)\}$ *corresponds to all occurrences* $\varphi = (e, \cdot)$ *of event* e *within trace* τ_S *on process schema* S*.*

For each time pattern we provide a description using the aforementioned schema (cf. Fig. 4 - Fig. 12). Additionally, we define pattern semantics by characterizing the traces τ_S that can be produced when executing any instance of process schema S while satisfying the time constraints expressed by the patterns.

Pattern Category I (Durations and Time Lags). Our first category comprises three time patterns expressing durations of process elements as well as time lags between them. Design Choice D constitutes a general design choice valid for all patterns from this category. It describes whether time lags are specified in terms of minimum/maximum values or time intervals (cf. Fig. 3).

General Design Choice for Pattern Category I
D.) There are three kinds of restrictions
a.) Minimum value,
b.) Maximum value and
c.) Time interval [min … max]

Fig. 3. General Design Choices for Category I

[1] Including start and end events of the activities.

[2] We assume that events in τ_S do not occur at the exactly same point in time.

Pattern TP1 (*Time Lags between two Activities*). This pattern is described in Fig. 4. It enables definition of different kinds of time lags between two activities. Informally, semantics of pattern TP1 is as follows: A time lag between activities A and B can be mapped onto a time lag between their start/end events e_A and e_B. This way, for example, start/start as well as end/start time lags can be described (see Design Choice E in Fig. 4). Compliance of a given trace with such time lag now means that all occurrences $\varphi = (e_A, t_A)$ and $\psi = (e_B, t_B)$ of the two events fulfill the given time lag. Note that this also needs to be valid in connection with loops. As illustrated in Fig. 5, considering time lags, for which one of the activities resides inside a loop and the other one outside that loop (e.g. A1 and A3 or A3 and A6 in Fig. 5) is problematic due to unclear semantics (for details see [14]). This becomes even more complicated when considering nested loops. For example a time lag between activities A_1 and A_6 in Fig. 5 could relate to the first iteration, the last iteration, every iteration or any special iteration

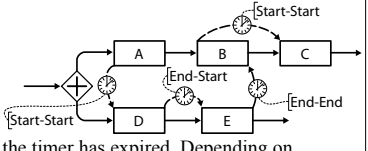
Time Pattern TP1: Time Lags between two Activities	
Also known as	Upper and Lower Bound Constraints, Inter-Task Constraints, Temporal Relations
Problem	There is a given time lag between two activities which needs to be respected. Time Lags may not only exist between succeeding activities, but also between arbitrary ones. Time lags are often required to comply with existing rules and regulations. The time lag may or may not have binding character.
Design Choices	D.) Time Lags may represent all three kinds of restrictions (cf. Fig. 3) E.) Time Lags can be realized based on four different time relations a.) Between start of two activities (i.e., Start-Start relation) b.) Between start of the first and completion of the second activity (i.e., Start-End) c.) Between completion of the first and start of the second activity (i.e., End-Start) d.) Between completion of two activities (i.e., End-End)
Solution	A time constraint is introduced between the start and / or end event of the two activities. Timers may be used to realize this pattern at runtime. For example, to realize an end-start relation, the timer starts after completing A. If the time lag between A and B is a minimum time lag, B may only be started after the timer has expired. Depending on whether a time lag has binding character the activation of the activity may be delayed until the time lag is satisfied. If the time lag is a maximum time lag B may be started as soon as the timer is started until its expiry. In case the timer expires an exception is raised. For time intervals both of the above cases apply. 
Context	The mechanism evaluating the constraint (i.e., starting the timer) needs to be able to access the value of the time lag when it determines the impact of the constraint.
Examples	• The *maximum time lag* between discharge of a patient from a hospital and sending out the discharge letter to the general practitioner of the patient should be 2 weeks (Design Choices D[b] E[d]) • Patients must not eat *at least 12 hours* before a surgery takes place. The latest point in time where the patient can have a meal is determined by the date of the surgery (Design Choices D[a] E[c]) • A contrast medium has to be administered *2 to 3 hours before* a radiological examination. The interval in which the contrast medium should be administered depends on the examination date (Design Choices D[c] E[a])
Related Patterns	TP2 – Durations TP3 – Time Lags between Events; TP1 can be implemented based on pattern TP3
Known uses	MS Project, BPMN, Eder et al. [2], Bettini et al. [4], Combi et al. [1]

Fig. 4. TP1 - Time Lags between Activities

of the outer loop and the inner one respectively. Hence this case needs to be excluded when defining semantics of pattern TP1.

To formally express this we define function $iteration(S, \varphi, \tau)$. It returns ordered set $(e_0 : n_0, e_{L_1} : n_{L_1}, \ldots, e_{L_k} : n_{L_k})$ which uniquely identifies each loop and its current iteration count with respect to a possibly surrounding loop (cf. Fig. 5). Thereby, e_0 is the start event of the respective process instance of schema S and e_{L_i}, $(1 \leq i \leq k)$ is the first event of a loop containing event e of event occurrence $\varphi = (e, \cdot)$[3] (e.g., e_2 and e_4 for start event $e_{A_6 S}$ of activity A_6 in Fig. 5). n_{L_i} $(1 \leq i \leq k)$, in turn, designates the iteration count of an inner loop e_{L_i} with respect to an outer loop $e_{L_{i-1}}$ (cf. Fig. 5), with n_0 always having value 1. A minimum time lag t_{min} (Design Choice D[a] in Fig. 3), for example, can now be formalized as follows:

$$\forall \varphi \in occurrences(S, e_A, \tau) \forall \psi \in occurrences(S, e_B, \tau) :$$
$$iteration(S, \varphi, \tau) = iteration(S, \psi, \tau) \Rightarrow \varphi^t + t_{min} \leq \psi^t.$$

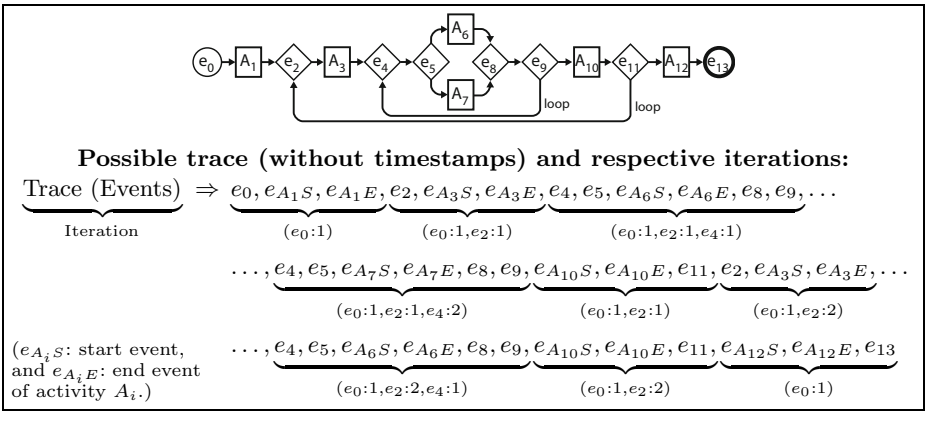


Fig. 5. Nested Loops and Iterations

Pattern TP2 (*Durations*) is described in Fig. 6. It allows specifying duration of process elements. If pattern TP2 is applied to an activity or process, same compliance rules as for pattern TP1 must hold. Thereby, e_A corresponds to the start event of the respective activity (process), while e_B corresponds to its end event. For a set of activities (process instances) (see Design Choices C[b] and C[d] in Fig. 2b), in turn, these rules need to be applied to the first start event and the last end event of all activity (process) instances of this set.

Pattern TP3 (*Time Lags between arbitrary Events*). TP3 is described in Fig. 7. It enables specification of time lags between two discrete events. Semantics of this pattern is similar to the one of TP1. However, no restrictions regarding events e_A and e_B apply (i.e., respective events do not need to be start/end events of activities). Thus, opposed to TP1, TP3 provides more generic support

[3] Here we only consider well-nested loops.

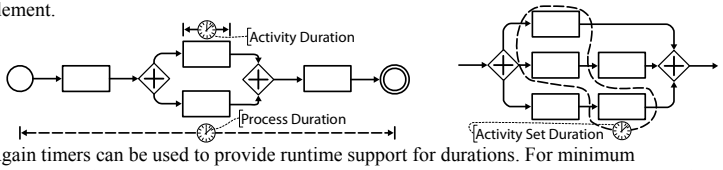
Time Pattern TP2: Durations	
Also known as	-
Problem	A particular process element has a certain duration restriction. Durations result from both waiting and processing times. Durations are often determined by external benchmarks (e.g., regulations, policies, QoS agreements). The duration may or may not have binding character.
Design Choices	C.) Durations can be applied to all four kinds of process elements (cf. Fig. 2b) D.) Durations may represent all three kinds of restrictions (cf. Fig. 3)
Solution	A time constraint is introduced between the start and end event of the particular process element.  Again timers can be used to provide runtime support for durations. For minimum (maximum) durations the respective element must not complete before (after) the timer has expired, otherwise appropriate exception handling is initiated. For intervals, the completion event has to occur within the interval boundaries.
Context	The mechanism evaluating the constraint (i.e., starting the timer) needs to be able to access the value of the duration before the particular element is executed.
Examples	• The assembly of a new engine must not take longer than 30 minutes (task work) (Design Choices C[a], D[b]) • Depending on its severity, ovarian cancer surgeries take 1 to 10 hours (Design Choices C[a], D[c]). • Maintenance issues need to be resolved within 1hr (Design Choices C[c], D[b]) • Processing 100 requests must not take longer than 1 second (Design Choices C[d], D[b])
Related Patterns	TP1 – Time Lags between Activities TP3 – Time Lags between Events – TP2 can be implemented based on TP3
Known uses	MS Project, BPMN, MQ Workflow, Eder et al. [2], Bettini et al. [4], Combi et al. [1]

Fig. 6. TP2 - Durations

for expressing arbitrary time lags. For example, respective events can be triggers from an external source (e.g., receiving a message, occurrence of a heart stroke) not controllable by the PAIS. In addition, they may refer to events not bound to a specific activity (e.g., event "delivery of all parts" requires several activities/processes to complete) or to events triggered inside an activity (e.g., milestone of an activity or subprocess, occurrence of exceptions).

Pattern Category II (Restrictions of Process Execution Points). This category comprises four patterns for restricting execution points (e.g., earliest start or latest end time) of process elements. Regarding this category design choice F describes what kind of execution point is specified by the respective constraint (e.g., earliest start or latest end date) (cf. Fig. 8).

Pattern TP4 (*Fixed Date Element*). TP4 is described in Fig. 9. It provides support for specifying a deadline. In many cases, fixed date elements implicitly determine latest (earliest) start (end) time of preceding (succeeding) activities as well. In most cases, the value of such fixed date element may not only depend on process schema S, but also on the current process instance (i.e., trace τ) and current iteration I of respective process element A. Therefore, we define function

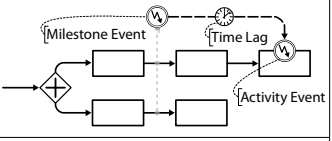
Time Pattern TP3: Time Lags between Arbitrary Events	
Also known as	-
Problem	There is a given time lag between two discrete events which needs to be respected. Events occur, for example, when instantiating or completing a process instance, when reaching a milestone in a process instance, or when triggering specific events inside an activity. Time lags are often required to comply with existing rules and regulations. The time lag may or may not have binding character.
Design Choices	D.) Time Lags between Events may represent all three kinds of restrictions (cf. Fig. 3)
Solution	A time constraint is introduced between the respective events. Again timers can be used to realize this pattern at runtime (cf. Fig. 4). Additionally an observer monitoring external events and notifying the mechanism evaluating the constraint is necessary. 
Context	The mechanism evaluating the constraint (i.e., starting the timer) needs to be able to access the value of the time lag in order to determine the impact of the constraint.
Examples	• Maximum time lags in an electronic change management process between sending a request for comments (by the partners affected by a change) and getting a response (Event) (Design Choices D[b]). • The time lag between delivery of all parts (milestone) and the assembly of the car's chassis (milestone) should be no more than 2 hours (e.g. just-in-time production) (Design Choices D[c]).
Related Patterns	TP1 – Time Lags between Activities TP2 – Durations
Known uses	Bettini et al. [4], Combi et al. [1]

Fig. 7. TP3 - Time Lags between Events

General Design Choice for Pattern Category II
F.) Patterns can restrict three dates of a process element a.) Earliest start date, b.) Latest start date, c.) Latest completion date

Fig. 8. General Design Choices for Category II

$fde(S, A, I, \tau)$, which returns for each process element A with fixed date element and each iteration I the current value of the fixed date element. Therefore fde effectively represents the Fixed Date attached to each Fixed Date Element (cf. Fig. 9). Compliance of a trace then means that each occurrence of the respective event e of element A in τ complies with the associated value of the fixed date constraint.

Pattern TP5 (*Schedule Restricted Element*). TP5 is described in Fig. 10. It enables us to restrict the execution of a particular element by a schedule; i.e., a timetable (e.g., a bus schedule). A particular schedule s_A attached to an activity (process) A can be instantiated as a (possibly infinite) set of subsets of the time points of a calendar C, i.e., $s_A \subseteq 2^C$. Depending on Design Choice G (cf. Fig. 10) the schedule is either instantiated as set of discrete time points $s_A = \{t | t \in C\}$ or as set of intervals $s_A = \{[t_{min}, t_{max}] | [t_{min}, t_{max}] \subseteq C\}$. An exception in respect to this schedule (cf. Fig. 10) is then expressed by removing and/or adding the respective time points or time intervals from/to the

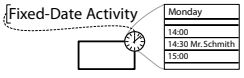
Time Pattern TP4: Fixed Date Elements	
Also known as	Deadline
Problem	A particular element has to be executed at a particular date. Fixed Date Elements often determine the latest or earliest start / completion time of preceding / succeeding activities as well. If the deadline is missed the activity or process may even become obsolete.
Design Choices	C.) A fixed date can be applied to an activity (a.) or process instance (c.) (cf. Fig. 2b) F.) A fixed date can restrict all three types of dates (cf. Fig. 8)
Solution	A Fixed Date is attached to the respective element. A Fixed Date can be realized using a timer which is started, as soon as the value of the fixed date is known and expires at the respective date. If, for example, for a latest start date the respective element has not been started before the timer has expired appropriate exception handling is initiated. This could, for example, lead to the cancelation of the respective activity. Other restriction can be handled analogously (cf. Fig. 4 for an example). 
Context	The value of the fixed date needs to be available prior to the respective activity becoming available for execution.
Examples	• Assume that software is released every two weeks on Friday evening. Thus, the deadline for changes (except bug fixes) is the day before the release date (time error might lead to delays or have no effect) (Design Choices C[a] F[c]). • To perform chemotherapy the physician has to inform the pharmacy about the dosage of the cytostatic drug until 11:00. If the deadline is missed the pharmacy checks back by phone for the exact dosage (escalation mechanism) (Design Choices C[a] F[c]). • A patient has an appointment for an examination Monday at 10:00, but due to a full schedule of the physician it may well be that the patient has to wait until the examination starts (i.e., earliest possible execution point is given) (Design Choices C[a] F[b]).
Related Patterns	TP5 – Schedule Restricted Elements; Fixed Date Elements are often schedule restricted elements as well.
Known uses	MS Project, BPMN, Eder et al. [2], Bettini et al. [4], Combi et al. [1]

Fig. 9. TP4 - Fixed Date Elements

schedule. To verify that a particular activity/process instance complies with the schedule it needs to be checked whether or not timestamp t of the respective event constitutes an element of the schedule (i.e. $t \in s_A$).

Pattern TP6 (*Time Based Restrictions*) enables us to restrict the number of times a particular process element can be executed within a predefined time frame (cf. Fig. 11). The particular time frame(s) are either defined by the time points of two events (Design Choice I[a]) or by a schedule (Design Choice I[b]). Based on these time frames it becomes possible to determine the number of executions within a particular time frame.

Pattern TP7 (*Validity Period*) enables us to restrict the lifetime of a process element to a given validity period (cf. Fig. 12). Semantics can be expressed by checking whether the timestamps of respective events lie within the particular validity period attached to the process element.

Pattern Category III (*Variability*) and **Pattern Category IV (*Recurrent Process Elements*).** Our catalogue comprises three additional patterns TP8, TP9 and TP10 covering time dependent variability, cyclic elements and periodicity. Due to lack of space we omit them here and refer to our technical report [14] instead.

5 Related Work

Patterns were first used by Alexander [15] to describe solutions to recurring problems and best practices in architectural design. Patterns also have a long tradition in computer science. Gamma et al. [16] applied same concepts to software engineering and described 23 design patterns. In the workflow area, patterns were introduced for analyzing expressiveness of process meta models [5,17]. In this context, control flow patterns describe constructs to specify activities and their ordering. In addition, workflow data patterns [6] provide ways for modeling the data aspect in PAIS. Furthermore, patterns for describing control-flow changes [18,7] and service interactions were introduced [19]. The introduction of workflow patterns has had significant impact on PAIS design and on the evaluation of PAIS and process languages. To evaluate powerfulness of a PAIS

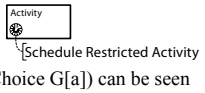
Time Pattern TP5: Schedule Restricted Elements	
Also known as	-
Problem	The execution of a particular element (i.e., activity or process) is restricted by a schedule. The structure of this schedule is known at process type level, while the concrete date is determined at instance level. The schedule provides restrictions on when the respective element can be executed. In particular, for rather restricted schedules even small delays in process execution can become critical (if schedule restricted elements being on a critical path are affected by the delay or the path becomes critical due to the delays). Schedules may contain exceptions (e.g., every year except leap years).
Design Choices	C.) A fixed date can be applied to an activity (a.) or process instance (c.) (cf. Fig. 2b) F.) A fixed date can restrict all three types of dates (cf. Fig. 8) G.) Execution of the element can be bound to a.) several discrete points in time (execution is only possible every full hour) or b.) one or more time frames (e.g. execution is only possible from 09:00 to 12:00)
Solution	A schedule is attached to the respective element.  A schedule restriction can be realized using a timer which is started when the process is started and expires when the first time frame of the schedule is reached (a discrete point in time (Design Choice G[a]) can be seen as a time frame with only one time point). The timer is then reset and its expiration date is set to the end of the next time frame of the schedule. This is repeated until no more time frames are in the schedule or the process element has been started / completed (cf. Design Choice F). If the start / end of the respective element does not occur within a valid time frame or there is no longer a time frame available in the schedule, appropriate exception handling is initiated.
Context	The schedule needs to be known at process type level or at least at process instantiation.
Examples	• Between Munich and Amsterdam there are flights at 6:05, 10:30, 12:25, 17:35 and 20:40 (Design Choice C[a] G[a]). • Opening hours of the dermatological clinic are MO – FR 8:00 – 17:00 except for public holidays. Dermatological examinations can only be scheduled within this time frame (Design Choices C[a] G[b]). • An information letter is sent by the leasing company to each customer within the first two weeks of each year (Design Choices C[a] G[b]) • Comprehensive lab tests in a hospital can only be done from MO – FR 8:00 – 17:00 (Design Choices C[a] G[b])
Related Patterns	TP4 – Fixed Date Elements (often schedule restricted elements) TP6 – Time Based Restrictions (like schedule based restrictions constrain possible execution points for an element) TP7 – Validity Period
Known uses	MS Project, Eder et al. [2], Combi et al. [1]

Fig. 10. TP5 - Schedule Restricted Element

regarding its ability to cope with time aspects, existing workflow patterns are important, but not sufficient. In addition, patterns addressing time constraints are needed.

Most academic approaches on time support for PAIS focus on time features like verification of time constraints [4,1,2], escalation management [9], and scheduling support [10,11]. The effect of ad-hoc changes on temporal constraints is investigated in [20]. A systematic investigation of requirements for time support from different heterogeneous application domains is missing so far.

6 Summary and Outlook

We have proposed time patterns to foster selection of appropriate PAIS-enabling technologies and to facilitate comparison of process management systems, calendar systems and project planning tools regarding their coverage of the time perspective in PAIS. In [14] we provide additional time patterns as well as a pattern-based evaluation of existing systems based on the time patterns. We

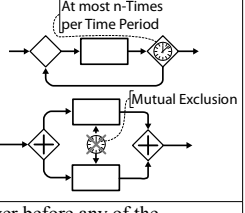
Time Pattern TP6: Time Based Restrictions	
Also known as	Some occurrences of this pattern are often referred to as "Mutual Exclusion"
Problem	Particular process elements may only be executed a limited number of times within a given timeframe. Time Based Restrictions are often needed to express the influence of resource restrictions (resource shortage) onto process execution.
Design Choices	H.) Time Based Restrictions can be applied to different types of process elements a.) Instances of single activity or group of activities within same process instance b.) Instances of single activity or group of activities within different process instances (potentially sharing some common characteristics) c.) Instances of a process or group of processes I.) There are two types of restrictions which can be expressed by Time Based Restrictions a.) Number of concurrent executions (at same time / with overlapping time frames) or b.) Number of executions per time period
Solution	To implement this pattern a constraint expressing a particular Time Based Restriction is associated with the process elements affected by this restriction. Additionally, the constraint specifies the respective time period and the number of executions. During runtime an observer can be used to monitor the number of running instances per time period and to raise an exception in case the maximum number of executions is exceeded. 
Context	The number of executions needs to be accessible by the observer before any of the respective process elements is started.
Examples	• Two invasive examinations must not be performed on same day (Design Choices I[b]). • For USD 19.90 10 different online books can be read per month. If the book tokens are consumed no more books can be read in the current month. At beginning of next month the book tokens get renewed (Design Choices H[a] I[b]). • During your stay at a wellness hotel you can select one treatment (free of charge) per day (Design Choices H[a] I[b]).
Related Patterns	TP5 – Schedule Restricted Elements; While the execution point of a schedule restricted element is constrained by a schedule, time based restrictions constrain the amount of activity instances / time period.
Known uses	-

Fig. 11. TP6 - Time Based Restrictions

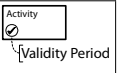
Time Pattern TP7: Validity Period	
Also known as	-
Problem	A particular process element may be only executed within a particular validity period, i.e., its lifetime is restricted to the validity period. The respective process element may only be instantiated within this validity period. In general, different versions of a process element may exist, but only one is valid at a specific point in time. Validity dates are especially relevant in the context of process evolution to restrict the remaining lifetime of an obsolete process implementation and to schedule rollout of the new process.
Design Choices	C.) A validity period can be applied to an activity (a.) or process instance (c.) (cf. Fig. 2b) F.) A validity period can restrict all three types of dates (cf. Fig. 8)
Solution	To realize this pattern a validity period is attached to the respective element. Upon instantiation of the respective process element, its validity period needs to be checked. If the element does not lie within its validity period or the duration of the element (see Fig. 5) leads to the end event being outside of the validity period, appropriate error handling is required. 
Context	The validity period needs to be known at process type level. If the validity period is bound to an activity it may apply to several different process types.
Examples	• Starting from Jan 1st patients need to be informed about any risks before the actual treatment takes place (Design Choice C[c] F[a]). • From next week on the new service version should get life (Design Choice C[a] F[a]). • Due to changed law, process A may only be used until January 1st. After this date no new process instances can be instantiated based on A, but process B has to be used instead (Design Choice C[c] F[b]).
Related Patterns	TP5 – Schedule Restricted Elements TP8 – Time Dependent Variability
Known uses	MQ Workflow

Fig. 12. TP7 - Validity Period

have shown that suggested time patterns are highly relevant in practice and complement existing workflow patterns with another fundamental dimension. In future work we will provide a reference implementation. Furthermore, we will conduct a comprehensive study of time support features (e.g., verification of time constraints, escalation management, scheduling support), in addition to the proposed time patterns, and also consider the resource dimension in this context.

References

1. Combi, C., Gozzi, M., Juarez, J., Oliboni, B., Pozzi, G.: Conceptual modeling of temporal clinical workflows. In: Proc. TIME 2007, pp. 70–81 (2007)
2. Eder, J., Panagos, E., Rabinovich, M.: Time constraints in workflow systems. In: Jarke, M., Oberweis, A. (eds.) CAiSE 1999. LNCS, vol. 1626, pp. 286–300. Springer, Heidelberg (1999)
3. Dadam, P., Reichert, M., Kuhn, K.: Clinical Workflows - The Killer Application for Process-oriented Information Systems?. In: Proc. BIS 2000, pp. 36–59 (2000)
4. Bettini, C., Wang, X.S., Jajodia, S.: Temporal reasoning in workflow systems. In: Distributed and Parallel Databases, pp. 269–306 (2002)
5. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow Patterns. Distributed and Parallel Databases 14, 5–51 (2003)
6. Russell, N., ter Hofstede, A., Edmond, D., van der Aalst, W.: Workflow Data Patterns. Technical Report FIT-TR-2004-01, Queensland Univ. of Techn. (2004)

7. Weber, B., Reichert, M., Rinderle-Ma, S.: Change Patterns and Change Support Features -Enhancing Flexibility in Process-Aware Information Systems. Data and Knoweldge Engineering 66, 438–466 (2008)
8. Russell, N., van der Aalst, W., ter Hofstede, A.: Exception Handling Patterns in Process-Aware Information Systems. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 288–302. Springer, Heidelberg (2006)
9. van der Aalst, W.M.P., Rosemann, M., Dumas, M.: Deadline-based escalation in process-aware information systems. Decision Support Systems 43, 492–511 (2007)
10. Combi, C., Pozzi, G.: Task scheduling for a temporal workflow management system. In: Proc. TIME 2006, pp. 61–68 (2006)
11. Eder, J., Pichler, H., Gruber, W., Ninaus, M.: Personal schedules for workflow systems. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 216–231. Springer, Heidelberg (2003)
12. Schultheiß, B., Meyer, J., Mangold, R., Zemmler, T., Reichert, M.: Designing the processes for chemotherapy treatment in a Women's Hospital (in German) (1996)
13. German Association of the Automotive Industry: Engineering Change Management. Part 1: Engineering Change Request, V 1.1., Doc. No. 4965 (2005)
14. Lanz, A., Weber, B., Reichert, M.: Time patterns in process-aware information systems - a pattern-based analysis - revised version. Technical Report UIB-2009, University of Ulm, Germany (2009)
15. Alexander, C., Ishikawa, S., Silverstein, M.: A Pattern Language. Oxford University Press, New York (1977)
16. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading (1995)
17. Puhlmann, F., Weske, M.: Using the Pi-Calculus for Formalizing Workflow Patterns. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 153–168. Springer, Heidelberg (2005)
18. Rinderle-Ma, S., Reichert, M., Weber, B.: On the formal semantics of change patterns in process-aware information systems. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 279–293. Springer, Heidelberg (2008)
19. Barros, A., Dumas, M., ter Hofstede, A.: Service Interaction Patterns. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 302–318. Springer, Heidelberg (2005)
20. Sadiq, W., Marjanovic, O., Orlowska, M.E.: Managing change and time in dynamic workflow processes. Int. J. Cooperative Inf. Syst. 9, 93–116 (2000)

On the Suitability of Aggregated and Configurable Business Process Models

Thomas Baier[1], Emilian Pascalau[2], and Jan Mendling[1]

[1] Humboldt-Universität zu Berlin, Germany
{Thomas.Baier,Jan.Mendling}@wiwi.hu-berlin.de
[2] Hasso Plattner Institute at the University of Potsdam, Germany
Emilian.Pascalau@hpi.uni-potsdam.de

Abstract. Reference models play an important role for specifying knowledge about a certain business domain that is general enough to be applicable for a wide set of companies. Still, it is understood that reference models need to be adapted in order to also reflect individual characteristics of a company. This adaptation turns out to be quite labor-intense. Concepts such as configurable process modeling languages have been proposed to simplify this adaptation. Competing languages have been designed to facilitate the actual act of adapting reference models, namely configurable EPCs (C-EPCs) and aggregated EPCs (aEPCs). In this paper we discuss the ease of use of these languages from an analytical perspective. Based on a mapping from C-EPCs to aEPCs we identify complexity issues and comparative advantages. It turns out that C-EPCs appear to be better suited to capture complex configuration options in a compact way.

1 Introduction

Reference models are a commonly used way to standardize and reuse existing knowledge about IT systems as well as business processes [1],[2]. As [3] states, reference models are mainly characterized by universality and a recommendation character. These aspects point to the need of different specializations of a reference model, e.g. for different countries or different industries. Therefore a reference model needs to be customized before it can be applied in the individual context of a certain company or organization. Rosemann and van der Aalst argue that common reference modeling languages like EPCs do not directly capture possible system configurations [4]. Different approaches have been proposed to overcome this problem, but a comparative analysis of their strengths and weaknesses is missing so far.

In [4] Rosemann and van der Aalst introduce a configurable reference modeling language, called Configurable EPCs (C-EPCs), as an extension to the widely used EPC modeling language. C-EPCs solve certain problems concerning configuration that arise from the limited ability of standard EPCs to distinguish between choices that can be made at runtime and choices that have to be made before, i.e. at configuration time [5]. For reference models represented using traditional process modeling languages this means that it is not possible to distinguish

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 108–119, 2010.

between mandatory elements and elements that could be omitted. Thus, it is not obvious if a choice within a model is meant to be made at the time the reference model is customized and therefore a certain part of the model is removed or if it is meant to be made at runtime. A different approach is taken in [6] where existing process models for different product variants are combined into one holistic model. It is designed as an extension to EPCs and called aggregated EPCs (aEPCs).

Up until now there is no research available on comparing aEPCs and C-EPCs. In this paper we investigate the mutual benefits of the two approaches from an analytical perspective. From such an investigation we aim to draw conclusions for future language design and give guidance for the selection of an approach in industry projects. We will define a mapping from C-EPCs to aEPCs to identify complexity issues and comparative advantages. The paper is structured accordingly. Section 2 introduces the main concepts of the two approaches using a working example. Afterwards Section 3 discusses a mapping from C-EPCs to aEPCs. Section 4 relates our research to similar work before Section 5 concludes the paper.

2 Adaptation of EPC Reference Models

In this section we discuss the adaptation of EPC reference models according to the two approaches. First, Section 2.1 introduces C-EPCs before Section 2.2 turns to aEPCs. We assume that the reader has some basic knowledge of EPCs or similar flow-charting diagrams like BPMN (see [7]).

2.1 Introduction to Configurable Process Models

Configurable process models have been defined to facilitate the adaptation of best practice reference process models by making configuration options explicit. Figure 1 shows the use of configurable process models in the process management lifecycle. The design phase is separated into two steps: first the reference model is built and second it is configured and tailored for the individual use case. For configurable process models, this means removing parts which are not relevant to the particular business use. The key idea is that the configurable process model would be typically provided by a standard software vendor and the individual company would not have to start from scratch. Figure 2 shows an example of a configurable process model using the C-EPC modeling language. It defines the process for call agent when a customer reports an incident. The process model depicts the four extensions that C-EPCs introduce for standard EPCs: `configurable connectors`, `configurable functions`, `requirements` and `guidelines` [8]. Configurable connectors and functions are highlighted with a bold border. They capture choices that have to be made when the model is configured. The `XOR` split in the example depicts such a choice. Depending on the companies requirements the `XOR` could, for instance, be configured to include only one of the shown sequences. In this case the company might only have a call

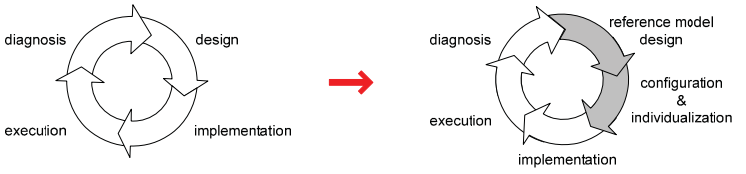


Fig. 1. Configuration and individualization of process models in the process lifecycle [5]

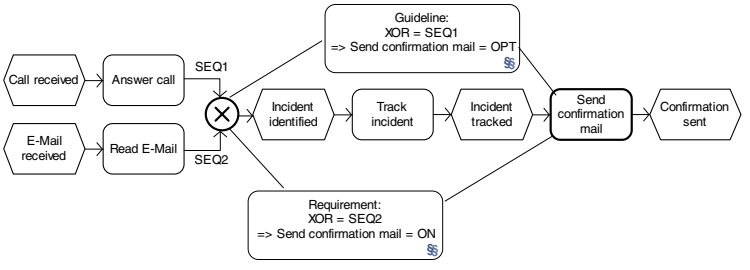


Fig. 2. C-EPC showing an excerpt of a configurable help desk process

center which is not capable of handling mails. Nevertheless, the `configurable XOR` could also be configured to a normal `XOR` indicating that both input channels are available and the agent has to make the choice at runtime. Thus, a `configurable XOR` connector can either be configured to reflect one of the possible sequences or to a normal exclusive choice [4].

Furthermore, the C-EPC in Fig. 2 includes a configurable function, the `"Send confirmation mail"` function. Configurable functions can be either included (`ON`), excluded (`OFF`) or conditionally skipped (`OPT`) [4]. In our example this means that we can globally decide at configuration time whether the agent always has to send a confirmation mail (`ON`) or not (`OFF`). Moreover, we could also leave the choice at the agent depending on the customers preferences and configure the function to be optional (`OPT`).

Besides the two already mentioned configurable elements, C-EPCs also allow to specify dependencies between certain configurations of configurable elements. In our example such a dependency is modeled with a so-called requirement. This requirement ensures that whenever we choose at configuration time to only have the mail channel as input, we always have to send confirmation mails. Another way to model dependencies is shown in Fig. 2 with a guideline which states that the `"Send confirmation mail"` function should be configured to be optional whenever we include only sequence 1, e.g. the telephone channel. In contrast to requirements, guidelines are only soft constraints that are not enforced. Hereby the modeler of a reference model can give advices to the person in charge of the configuration [9].

As there are often interdependencies between the choices in a configurable process model, foremost when models get bigger, La Rosa et al. [10] developed a

questionnaire-based approach to configure process models. This approach aims at preventing the user of a configurable model to make inconsistent choices. First, it allows to order the choices in a way that helps reducing the amount of inconsistent choices that can be made. Second, whenever a choice is made, all prohibited choices that might follow are pruned.

2.2 Introduction to Aggregated Process Models

Similar to C-EPCs, aggregated EPCs (aEPCs) have been developed to combine related process models into a single model. The idea is that an original singular process can be extracted for communication purposes [6]. The extraction of singular process models is necessary as aggregated processes tend to get too complex for communication. What is more, most stakeholders only need a particular part of a process and might get confused by too big models that only contain a small amount of relevant data for them. Aggregated EPCs therefore introduce a labeling concept to distinguish between parts which are common for a certain number of processes and parts which are uniquely dedicated to a particular process. By doing so the original singular process can be reconstructed.

In Fig. 3 the same process example as used above is shown in form of an aggregated EPC (aEPC). This aEPC aggregates the two processes for the handling of an incident that is reported via mail and for an incident reported via phone. All events and functions have been labeled either with "`Mail`", with "`Phone`" or with "`Incident`". The latter one describes all process parts that both processes have in common while the other two labels describe specific processes. In order to model the relations between these labels there is a tree hierarchy introduced as shown in Fig. 4 for our process example [6]. When we choose a label from this hierarchy to extract all belonging process parts, the extraction algorithm described in [6] will extract all parts labeled with this label plus all parts which are labeled with a label that is on a path through the chosen label starting from

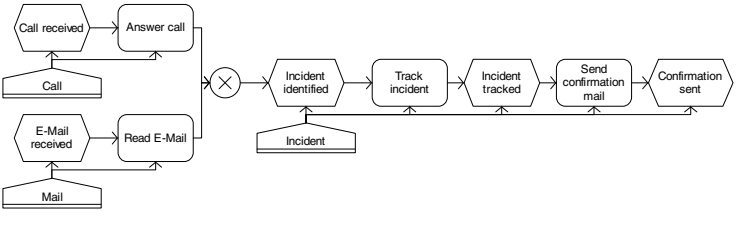


Fig. 3. aEPC showing an excerpt of a help desk process

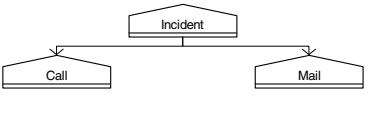


Fig. 4. Label hierarchy for the process model shown in Fig. 3

the hierarchy root to every leaf. Thus, we can extract exactly the two processes for using the mail and the phone channel without having to label every common part with the two specific labels "`Mail`" or "`Phone`".

3 Transformation between aEPCs and C-EPCs

Before we can assess strengths and weaknesses of C-EPCs and aEPCs, we discuss challenges in mapping C-EPCs to aEPCs in Section 3.1. Then, Section 3.2 draws conclusions from this discussion.

3.1 C-EPC to aEPCs Transformation Patterns

In this section we want to focus on the transformation of C-EPCs to aEPCs. We decided to focus on transforming C-EPCs because they provide different explicit configuration constructs in the language itself while aEPCs only use one main concept which is labeling. Thus, in the transformation from C-EPCs to aEPCs all constructs of both concepts can be covered. Therefore we need to look at the parts of a C-EPC which distinguish it from aEPCs. These parts are configurable functions, configurable connectors, requirements and guidelines. Concerning these modeling elements we will introduce basic informal transformation patterns which we have identified to perform the transformation of a C-EPC to an aEPC.

Configurable functions. Starting with the *pattern for configurable functions*, Fig. 5 depicts two process models. The first is a C-EPC with two configurable functions F2 and F3 while the second process model shows the first steps of a transformation to an aEPC. As mentioned before in section 2, a configurable function can either be `on`, `off` or `conditionally skipped`. To reflect this in an aEPC an XOR split with two outgoing arcs has to be inserted before every configurable function as shown in Fig. 5 (2) to indicate the choice. Both paths start with an event which indicates whether the function is `on` or `off`. The upper path includes the function F2 and the following event while the bottom path only consists of the event indicating that the function F2 is `off`. The latter event is necessary because without it the bottom path would always be included in every possible configuration. The construct is then finished with the XOR join

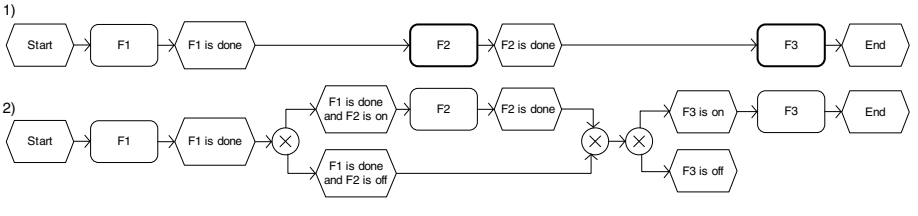


Fig. 5. Transformation of a C-EPCs with two configurable functions to an aEPC

which merges the two paths. As F2 is directly followed by another configurable function the described pattern immediately repeats. The XOR join in this case can be omitted because there is no function following F3 and hence, no join of the two paths is necessary. Thus both events `"End"` and `"F3 is off"` are now end events.

In order to obtain an aEPC with the same extraction results as the C-EPC, we first have to build a label hierarchy as shown in Fig. 6 and attach these labels to the corresponding events and functions. The label hierarchy therefore has to be built during the actual transformation process. For simplicity we assume that we have only one start event with which we can start and then recursively iterate through all elements. At first we create a root label which has to be attached to all parts that belong to every possible configuration. In our example this is the label `F` which is attached to the start event (`Start`), F1 and the event `"F1 is done"`.

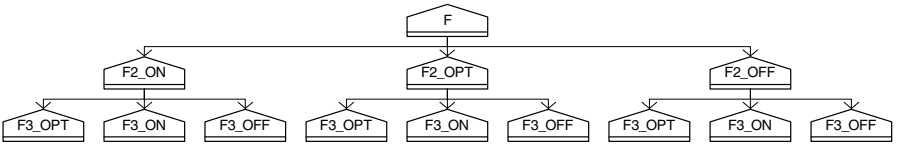


Fig. 6. Label hierarchy for the transformation of configurable functions shown in Fig. 5

Beginning with the start event we iterate through the C-EPC element by element. When we reach the first configurable function F2 in the C-EPC and create the described XOR split, we have to create a label for each possible configuration of F2: `"F2_ON"`, `"F2_OFF"` and `"F2_OPT"`. The label `"F2_ON"` is attached to all elements on the upper path and the label `"F2_OFF"` is attached to the event on the bottom path. The configuration that F2 can be optionally skipped is expressed by maintaining both paths. Therefore all elements on both paths are labeled with `"F3_OPT"`. Afterwards, the three labels have to be added to the hierarchy as direct children of the root label.

Proceeding with F3 we have to create three new labels `"F3_ON"`, `"F3_OFF"` and `"F3_OPT"` and attach them to the elements in the same way as described for F2. These three labels will then be added to every leaf of the label hierarchy as shown in Fig. 6.

The transformation in Fig. 5 (2) is not yet a valid aEPC as there are events which are directly followed by other events which is not allowed in an EPC (see [11]). In order to obtain a valid EPC it is necessary to introduce a skip function (see Fig. 7) as proposed by Rosemann and van der Aalst in [4]. Thus, we directly obtain a valid aEPC. The drawback of this solution is that we loose the predecessor events of the configurable functions.

Preserving the predecessor events of the configurable functions can be achieved by introducing an additional decision function Z as also proposed in [4]. This is shown in Fig. 8. Here we also introduce a new concept for the labeling. The first

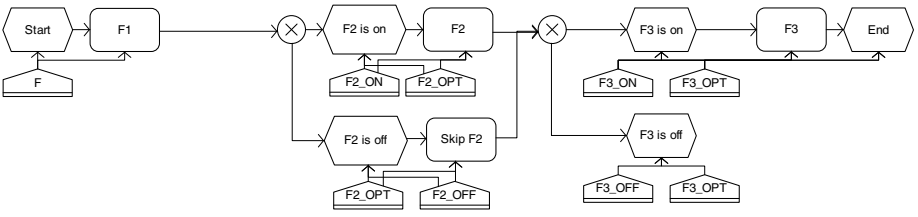


Fig. 7. aEPC using a skip function for the transformation of configurable functions

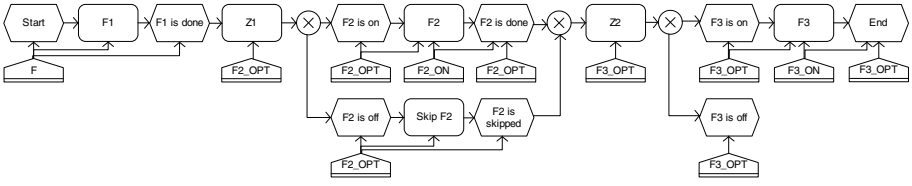


Fig. 8. aEPC using a skip function and a decision function `Z` for the transformation of configurable functions

decision function `Z1` is labeled only with `"F2_OPT"` and the events following that decision function are labeled only with that label, too. The skip function is also labeled only with `"F2_OPT"`. Note that thus, the label `"F2_OFF"` is not contained in the aEPC anymore. Nevertheless, it will remain in the label hierarchy. Using this labeling concept the extraction result of our aEPC will only include the additional decision and skip functions and the additional events when we select the label `"F2_OPT"` and not if we select the label `"F2_OFF"` or `"F2_ON"`. Hence, we do not have any unnecessary additional elements when configuring a function to be `on` or `off`. Thus, we achieve the same configuration results as obtained by the C-EPC following the proposals of handling optional functions by Rosemann and van der Aalst.

Configurable connectors. Figure 9 shows the transformation of a configurable OR connector (1) to an equivalent aEPC representation (2). A configurable OR connector can either be configured to an XOR, an OR, an AND or to a sequence choosing only one of the outgoing paths [6]. The choice between these configurations is represented by the XOR connector which follows function `A1` in Fig. 9 (2). Each possible configuration is represented in one of the outgoing paths of this XOR split and labeled with an individual label. The label hierarchy for this aEPC is very simple. The label `A1` is the root of the tree and all other labels that represent the possible configurations are leafs. Note that choosing the root of the tree always results in the whole aEPC which is not a possible result of the configuration of the original C-EPC.

The configurable OR connector is the connector with the most possible configurations and was therefore chosen as representative example for all configurable

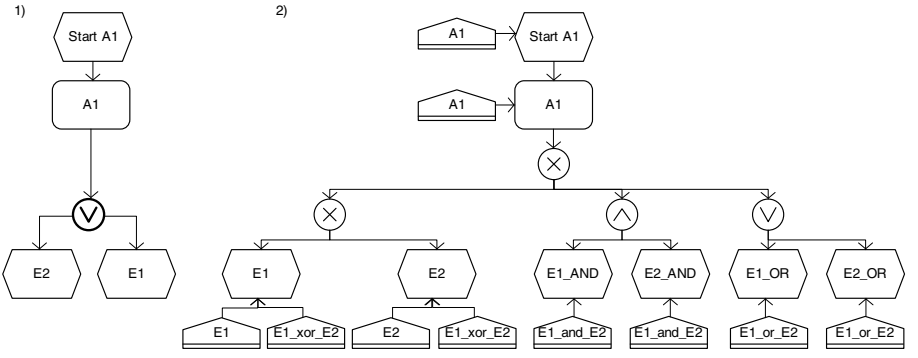


Fig. 9. Transformation of a C-EPC (1) with a configurable OR connector to an aEPC (2)

connectors. The AND and XOR connectors can be transformed in the same way and have only a subset of the possible configurations of the OR connector [4].

Requirements and Guidelines. Figure 10 (1) shows a C-EPC with two configurable functions `F1` and `F2` and a requirement enforcing `F2` to be configured `ON` if `F1` is configured `ON`. The configurable functions are translated similarly to the approach discussed before in this section using decision functions and skip functions. The only difference is that they are not in sequence anymore but in parallel. To ensure the requirement given in the C-EPC we just have to drop all configurations from the label hierarchy which are conflicting with this requirement. Therefore we

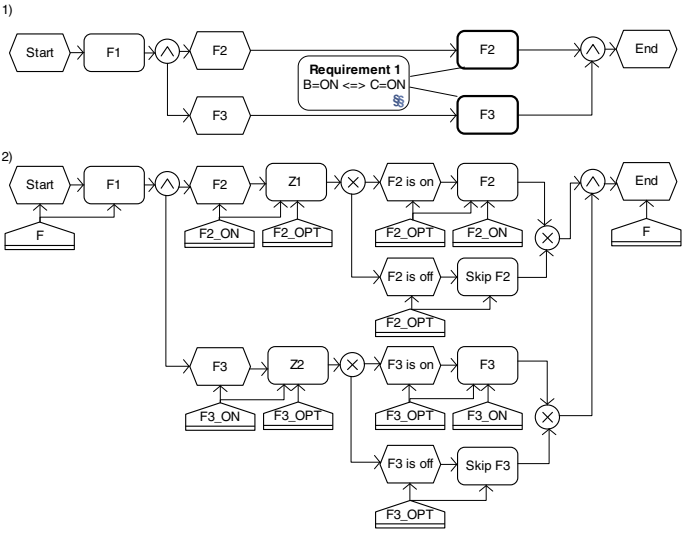


Fig. 10. Transformation of a C-EPC (1) with a requirement to an aEPC (2)

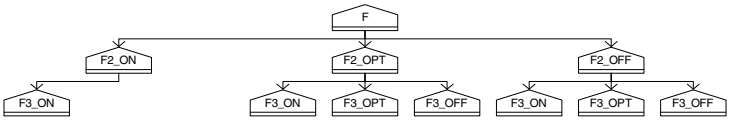


Fig. 11. Label hierarchy for the aEPC shown in Fig. 10

search the hierarchy for the equivalent label for the first term of the requirement. The term of the requirement is `"F2=ON"` and its equivalent configuration label is `"F2_ON"`. We then have to look at all possible paths that go from the root through the node `"F2_ON"` and contain a configuration for `F3` which is not the configuration enforced by the requirement. Hence, every configuration for `F3` on any path through the node `"F2_ON"` that does not configure `"F3=ON"` has to be dropped. Figure 11 shows the resulting label hierarchy.

Guidelines are defined in the same way in C-EPCs as requirements, but they only refer to soft constraints. Thus, they are only suggestions that the model customizer can interpret as a hint, but does not have to follow. In aEPCs there is no way to model those soft constraints as everything that is modeled explicitly influences the extraction. Hence, guidelines could only be provided in form of additional text separately from the model.

3.2 Discussion

The mapping patterns presented above point to some strengths and weaknesses of the two approaches. We focus on the visualization of configuration options, the navigation through options, and the visualization of dependencies.

Visualization of Configuration Options. The mappings highlight that transformation of configurable functions and connectors comes down to multiplying out all possible options. Therefore, C-EPCs offer a much more compact way of visualizing configuration options as each option is captured locally at a single node.

Navigation through Options. The aEPC approach, on the other hand, offers a useful means to navigate through different configuration options using the tree. The choice for a particular tree leaves yields a complete adaptation of the process model.

Visualization of Dependencies. Both approaches are rather weak in visualizing dependencies of adaptations. In a C-EPC dependent configurable nodes are linked to the same requirement although the direction of dependency is not visualized. In an aEPC forbidden options are simply skipped from the tree, but the causality is not depicted.

Altogether, C-EPCs offer are more compact visualization of adaptation options than aEPCs. This is beneficial when process models are large and capture several independent configuration options. aEPCs on the other hand help to navigate through different individualizations of a more generic model. This generic aEPC

model gets overly complex when an extensive set of adaptation options need to be described.

4 Related Work

In this paper, we have focussed on two extensions of EPCs for addressing requirements of reference models that relate to variants. The management of variants has been discussed in various domains before, among others in work on software configuration management [12,13,14] and feature diagrams [15,16,17]. For process models, different configuration approaches have been defined. We discuss configurable nodes, model projections, annotation approaches, and hiding and blocking, and relate them to C-EPCs and aEPCs.

C-EPCs utilize *configurable nodes* as a means of introducing configurability to EPCs. All configuration decisions have to be made on the level of nodes by setting a respective configuration value. This concept has been recently extended beyond control flow nodes [18]. aEPCs work on the principle of *projection*. Only those elements that have a particular label are included, the others are not included. The idea of projection is introduced in [19]. In their approach, configuration parameters can be set for elements of a model, but also for elements of the metamodel. In this way, element types can be excluded. A respective individualization algorithm [20] (pp. 141–142) is executed to remove the hidden elements and reconnect the remaining nodes. In [21] the authors show how this approach has been implemented as a toolset that builds on ARIS.

Different approaches have utilized *annotations* for configuration purposes. The PESOA (Process Family Engineering in Service-Oriented Applications) project [22,23] defines so-called *variant-rich process models* as process models extended with stereotype annotations to accommodate variability. These stereotypes are applied to both UML Activity Diagrams and BPMN models. The places of a process model where variability can occur are marked as variation points with the stereotype "VarPoint". These can be activities in UML Activity Diagrams and tasks in BPMN. Further stereotypes including "Variant", "Default", "Abstract", "Alternative", "Null", and "Optional" are used for the specification of different configuration options. A subset of these stereotypes proposed by the PESOA project appears in [24]. Other annotation approaches are defined in [25,26].

Work on configurable workflow languages uses skipping and blocking of tasks to configure control flow [27,28]. The concept is formalized using *hiding* and *blocking* operators from workflow behavior [29]. According to the inheritance of workflow behavior, *blocking* corresponds to disabling a particular activity and its subsequent path, while *hiding* corresponds to making an activity unobservable. A nice feature of this approach is that for free-choice Petri nets, the soundness correctness criterion is preserved no matter which configuration is chosen [30,31].

5 Conclusion and Outlook

Several reference model adaptation approaches have been defined, but a comparative analysis has been missing so far. In this paper we have addressed this lack

by focussing on C-EPCs and aEPCs. We discussed a mapping from C-EPCs to aEPCs to understand the mutual strengths and weaknesses of the two languages. We found that C-EPCs are more compact in capturing complex choices while aEPC variants can be easily navigated using the tree of the label hierarchy.

Our findings point to a need for further research on the actual procedure of adapting reference process models. Empirical research methods like think-aloud techniques and experiments might be a suitable means to investigate challenges in working with reference models from a modelers perspective. Furthermore, we have pointed out that dependencies are not directly visible in both languages, while C-EPCs are more explicit in this regard than aEPCs. It is a subject of future research to investigate ways of representing dependencies in a more readable way.

References

1. Küster, J.M., Koehler, J., Ryndina, K.: Improving Business Process Models with Reference Models in Business-Driven Development. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 35–44. Springer, Heidelberg (2006)
2. Fettke, P., Loos, P.: Classification of reference models: a methodology and its application. Information Systems and E-Business Management, 35–53 (2003)
3. Thomas, O.: Understanding the Term Reference Model in Information Systems Research: History, Literature Analysis and Explanation. In: Bussler, C.J., Haller, A. (eds.) BPM 2005. LNCS, vol. 3812, pp. 484–496. Springer, Heidelberg (2006)
4. Rosemann, M., Aalst, W.: A configurable reference modelling language. Inf. Syst. 32(1), 1–23 (2007)
5. La Rosa, M., Marlon, D.: Configurable Process Models: How To Adopt Standard Practices In Your How Way? (2008)
6. Reijers, H., Mans, R., Toorn, R.: Improved model management with aggregated business process models. Data Knowl. Eng. 68(2), 221–243 (2009)
7. Mendling, J.: Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness. LNBIP, vol. 6 (2008)
8. Aalst, W., Dreiling, A., Gottschalk, F., Rosemann, M., Jansen-Vullers, M.: Configurable Process Models as a Basis for Reference Modeling. In: Bussler, C.J., Haller, A. (eds.) BPM 2005. LNCS, vol. 3812, pp. 512–518. Springer, Heidelberg (2006)
9. Kindler, E., Nüttgens, M. (eds.): Business Process Reference Models. In: Proceedings of the Workshop on Business Process Reference Models, BPRM 2005 (2005)
10. La Rosa, M., Aalst, W., Dumas, M., Ter hofstede, A.H.M.: Questionnaire-based variability modeling for system configuration. Software and Systems Modeling (2008)
11. Aalst, W.: Formalization and Verification of EPCs (1999)
12. Estublier, J., Casallas, R.: The Adele Software Configuration Manager. In: Tichy, W. (ed.) Configuration Management. Trends in software. John Wiley & Sons, Chichester (1994)
13. Tryggeseth, E., Gulla, B., Conradi, R.: Modelling Systems with Variability using the PROTEUS Configuration Language. In: Estublier, J. (ed.) ICSE-WS 1993/1995 and SCM 1993/1995. LNCS, vol. 1005, pp. 216–240. Springer, Heidelberg (1995)
14. Turkay, E., Gokhale, A., Natarajan, B.: Addressing the Middleware Configuration Challenges using Model-based Techniques. In: Yoo, S., Etzkorn, L. (eds.) Proceedings of the 42nd ACM Southeast Regional Conference, pp. 166–170. ACM Press, New York (2004)

15. Batory, D., Geraci, B.: Composition Validation and Subjectivity in GenVoca Generators. IEEE Transactions on Software Engineering 23(2), 67–84 (1997)
16. Czarnecki, K., Helsen, S., Eisenecker, U.: Formalizing Cardinality-Based Feature Models and Their Specialization. Software Process: Improvement and Practice 10(1), 7–29 (2005)
17. Schobbens, P.Y., Heymans, P., Trigaux, J.C.: Feature Diagrams: A Survey and a Formal Semantics. In: Glinz, M., Lutz, R. (eds.) Int. Conf. on Req. Eng., pp. 136–145 (2006)
18. La Rosa, M., Dumas, M., Hofstede, A., Mendling, J., Gottschalk, F.: Beyond Control-Flow: Extending Business Process Configuration to Roles and Objects. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 199–215. Springer, Heidelberg (2008)
19. Becker, J., Delfmann, P., Dreiling, A., Knackstedt, R., Kuropka, D.: Configurative Process Modeling – Outlining an Approach to increased Business Process Model Usability. In: Khosrow-Pour, M. (ed.) Proceedings of the 14th Information Resources Management Association International Conference, pp. 615–619. IRM Press (2004)
20. Delfmann, P.: Adaptive Referenzmodellierung. Methodische Konzepte zur Konstruktion und Anwendung wiederverwendungsorientierter Informationsmodelle. Logos (2006) (in German)
21. Delfmann, P., Janiesch, C., Knackstedt, R., Rieke, T., Seidel, S.: Towards Tool Support for Configurative Reference Modeling – Experiences from a Meta Modeling Teaching Case. In: Brockmans, S., Jung, J., Sure, Y. (eds.) Proceedings of the 2nd International Workshop on Meta-Modelling. LNI, vol. 96, pp. 61–83 (2006)
22. Puhlmann, F., Schnieders, A., Weiland, J., Weske, M.: Variability mechanisms for process models. PESOA-Report 17/2005, Hasso-Plattner-Institut (June 2005)
23. Schnieders, A., Puhlmann, F.: Variability Mechanisms in E-Business Process Families. In: Abramowicz, W., Mayr, H. (eds.) Proceedings of the 9th International Conference on Business Information Systems. LNI, vol. 85, pp. 583–601 (2006)
24. Razavian, M., Khosravi, R.: Modeling Variability in Business Process Models Using UML. In: Latifi, S. (ed.) Proceedings of ITGN 2008, pp. 82–87 (2008)
25. Reinhartz-Berger, I., Soffer, P., Sturm, A.: A domain engineering approach to specifying and applying reference models. In: Desel, J., Frank, U. (eds.) EMISA 2005. LNI, vol. 75, pp. 50–63 (2005)
26. Czarnecki, K., Antkiewicz, M.: Mapping Features to Models: A Template Approach Based on Superimposed Variants. In: Glück, R., Lowry, M.R. (eds.) Proc. of the 4th Int. Conf. on Generative Programming and Component Eng., pp. 422–437 (2005)
27. Gottschalk, F., Aalst, W., Jansen-Vullers, M.: Configurable Process Models – A Foundational Approach. In: Becker, J., Delfmann, P. (eds.) Reference Modeling, pp. 59–78. Springer, Heidelberg (2007)
28. Gottschalk, F., Aalst, W., Jansen-Vullers, M.H., La Rosa, M.: Configurable workflow models. Int. J. Cooperative Inf. Syst. 17(2), 177–221 (2008)
29. Aalst, W., Basten, T.: Inheritance of Workflows: An Approach to Tackling Problems Related to Change. Theoretical Computer Science 270(1-2), 125–203 (2002)
30. Aalst, W., Dumas, M., Gottschalk, F., Hofstede, A., La Rosa, M., Mendling, J.: Correctness-preserving configuration of business process models. In: Fiadeiro, J.L., Inverardi, P. (eds.) FASE 2008. LNCS, vol. 4961, pp. 46–61. Springer, Heidelberg (2008)
31. Aalst, W., Dumas, M., Gottschalk, F., Hofstede, A., La Rosa, M., Mendling, J.: Preserving correctness during business process model configuration. Formal Aspects of Computing, 1–24 (2009)

Identifying Drivers of Inefficiency in Business Processes: A DEA and Data Mining Perspective

Anne Dohmen and Jürgen Moormann

Frankfurt School of Finance & Management, ProcessLab, Sonnemannstr. 9-11,
60314 Frankfurt am Main, Germany
{a.dohmen,j.moormann}@frankfurt-school.de

Abstract. Measuring the performance of business processes in the financial services sector can be tackled from different perspectives. The viewpoint of efficiency is one of them. This paper focuses on the analysis of process efficiency and proposes a new methodology for measuring process efficiency and for further identifying drivers of process inefficiency. It is suitable for a specific perspective on process efficiency. The methodology is based on Data Envelopment Analysis (DEA) and methods from Data Mining. It aims to find strong association rules between process transactions' characteristics and inefficiency values. This approach enables the identification of drivers of inefficiency from a (large) dataset of transactions without any prior assumptions about potential determinants of inefficiency. The methodology is applicable to business processes supported by workflow management systems and it can serve as the basis for an add-on system allowing structural analysis of process inefficiency and its drivers.

Keywords: Process Efficiency, Data Envelopment Analysis, Data Mining, Efficiency Drivers.

1 Introduction

Establishing efficient business processes is a goal companies in every industry should strive for. Especially in the banking sector, it is still a long way to go to achieve satisfactory process efficiency. This indicates that identifying drivers of inefficiency in this industry seems to be a worthwhile topic to be elaborated on. As a consequence we tackle the research question of our paper from the perspective of banking, even though the presented methodology might be applicable to other industries as well. A striking question arising in the context of efficiency in banking is: Why is it so difficult for financial service providers to establish efficient business processes, whereas other industries, such as the manufacturing sector, are years ahead in terms of process improvement? One reason might be the specific characteristics of services compared to physical products, so that classical production principles often cannot be applied one-to-one to service delivery processes [1].

In this paper we will not look at the problem of service delivery processes per se. Instead, we take into account production-like processes in financial services – in particular in banking – typically occurring in the back office. The focus will be set on a specific problem that might account for the disability for banks to establish efficient

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 120–132, 2010.

business processes even for highly standardized processes: Managers in banking operations often cannot identify the degree of inefficiency within a process and the reasons why these inefficiencies occur.

Until now, there does not exist any common and methodologically founded method for measuring process efficiency and for identifying potential drivers of inefficiency, which could support operations managers in analyzing process performance. Under the assumption that a clear understanding of the magnitude and the reasons for inefficiency in business processes is the first step for establishing high performance processes, this paper provides a promising procedure for measuring and identifying drivers of inefficiency. It is based on non-parametric methods, namely Data Envelopment Analysis (DEA) and methods from the field of Data Mining.

Different perspectives on process efficiency require different approaches for measurement and analysis. In this paper we look at a specific perspective on process efficiency and we emphasize that the approach we present is explicitly applicable for this perspective. It might have to be adjusted or replaced for any other perspective on process efficiency, which is regarded to be out of scope for this paper.

The paper is structured as follows. In section 2 the perspective on business process efficiency is defined and motivated. Section 3 includes the research question of the paper, the identified research gap, and related work. In section 4 our approach for measuring and identifying inefficiency drivers in business processes is described, followed by the conclusion and outlook in section 5.

2 Perspectives on Business Process Efficiency

As the very first step of analysis, the concept of efficiency has to be defined. Commonly, in practice it is understood as a rather loose concept, following citations like "efficiency is doing things right" or "efficiency is the relation of output to input". However, following the perspective of production theory, this is not exactly right. According to [2] efficiency in general is the comparison of a target value and the amounts of inputs required to achieve this target. From the perspective of production theory, objects classified to be efficient generate an empirical production function, representing the "real" objects which utilize minimal input for a given output or maximal output for a given input [2]. These are also called best practice objects. The inefficiency of the remaining objects is determined by their distance from this best practice frontier. In academic literature different types of efficiency exist, two of them are technical efficiency and allocative efficiency [3]. Input allocative efficiency is e.g. defined by [3, p. 27] as: "Input allocative efficiency reflects the 'distance' of the input mix used by an [object] from the optimal mix it could have used to minimize the cost of output, in light of input prices". This means that input allocative efficiency takes explicitly the factor prices of inputs into account, whereas output allocative efficiency would include the prices of outputs. Technical efficiency on the other hand takes into account the optimal combination of the *amount* of inputs and outputs without regarding any factor prices. The overall efficiency is generally considered to take into account both the optimal amount and factor prices of inputs [4]. Since in our paper we focus on the efficiency from a production perspective, we look at the technical efficiency. However an extension to allocative efficiency would be interesting if the relevant data for input factor prices is available.

Furthermore, it has to be defined which inputs and outputs should be selected in order to measure efficiency. Here, the perspective chosen has a substantial influence on efficiency measures and might distort the comparability of results. In particular, there is a distinction between modeling efficiency for banks from the *intermediation* or the *production* point of view [4]. In the intermediation view, inputs are generally deposits, whereas outputs are loans issued. From the perspective of production, physical resources, such as labor, are treated as inputs in order to generate transactions processed as output. This paper explicitly focuses on the production point of view. The production of banks is based on their business processes. Efficiency enhancements are especially important for standardized value creation which can typically be found in back office processes [5].

Generally, process performance can be improved in two ways: First, the process-flow can be changed; and second, the quality of the process execution can be improved by minimizing variations in the process performance for a given process design [6]. Whereas most papers deal with process design, this paper focuses on process execution, i.e. the intrinsic inefficiency of the process [7].

Figure 1 exhibits a hierarchical classification of the perspective on business process efficiency taken in this paper.

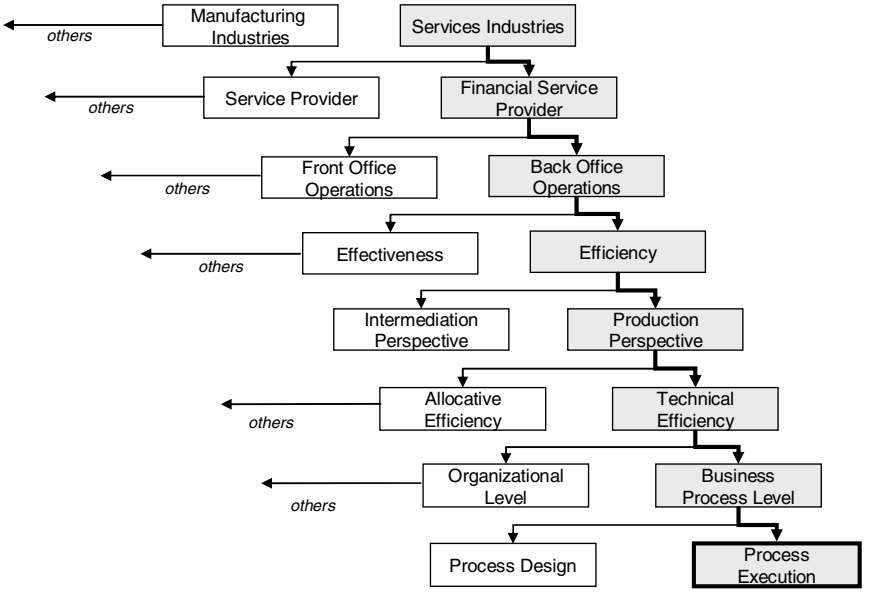


Fig. 1. Perspective on business process efficiency in this paper

3 Related Work and Research Design

Following the argumentation in the previous sections, the research question for this paper is formulated as follows: *How to measure and identify drivers of the intrinsic*

technical inefficiency in business processes of banking back office operations, following the production perspective on banking efficiency?

Since answering the research question has to consider previous findings from different disciplines, the following paragraphs will deliver a short overview on related work.

3.1 Related Work on Efficiency in Business Process Management

One stream of research in business process management includes developing systems and metrics for process performance analysis. Critical in this context is the use of simple metrics or ratios that usually fall short in capturing performance in its entirety [8]. Another stream of research deals with the development of process maps or formal process models as a basis for process analysis [9]. However, these approaches are more suitable for applying a perspective of process design analysis rather than for analyzing the efficiency in process execution. As a consequence, it seems to be more appropriate for measuring the intrinsic efficiency to focus on methods applying input-output models in order to assess the efficiency independent from the design of throughput [7]. These kinds on input-output models are utilized in traditional efficiency analysis based on frontier analysis, often being applied for measuring the efficiency of organizations in the financial services sector [10].

3.2 Related Work on Efficiency Measurement in the Financial Services Sector

Efficiency analysis in financial services, and in particular in banking, has a long tradition in academic literature. Predominantly, frontier analyses are applied, including parametric and non-parametric approaches [10]. Overall, Data Envelopment Analysis (DEA) – a nonparametric approach – came up as the most frequently used approach for measuring efficiency in banking [4].

DEA follows the definition of efficiency from a production theory perspective. Therefore, the approach for measuring the process efficiency presented here is based on this method. Following DEA, the efficiency measurement for the object of analysis (Decision Making Unit, DMU) is accomplished via the construction of an empirically based best practice production frontier and the evaluation of each DMU against a peer object on that frontier [11]. The distance from the best practice frontier determines the level of inefficiency. DEA allows for an efficiency measurement of multidimensional problem settings on the basis of very limited assumptions. DEA is especially appropriate for analyzing input-output relations in cases where the design of the transformation process and the production function are unknown [12].

A comprehensive overview on studies on efficiency measurement in the financial service sector using frontier analysis can e.g. be found in [10]. Until now, these studies have been performed on a high level of aggregation, namely on an organization's or at least branch's level [13]. Only very few studies can be found that apply frontier analysis to process level [6], but these studies suffer from methodological constraints [14]. The starting point for measuring the efficiency of banks should be at business processes level. The analysis of processes is the key element for evaluating service capability or performance [15].

Having identified this lack of research, [7] proposes a DEA-based method for measuring the intrinsic process efficiency. In this approach, called *Benchmarking of Transactions*, single transactions (the objects or process instances passing through the process) serve as DMUs in a Data Envelopment Analysis. Applying this method yields an empirical production function of best practice transactions and the transactions' inefficiency is determined by their distance from this frontier. As a result, an average efficiency score for the process can be calculated. This induces that the average efficiency of a large dataset of transactions from the same process gives an indication about the efficiency of the process or subprocess. In addition, benchmarking of transactions overcomes previous drawbacks of efficiency measurement techniques in business process management. Instead of being based on simple metrics, this method allows quantitative measurement of process efficiency from the perspective of production theory. We conclude that the concept of benchmarking of transactions following [7] is an appropriate method for measuring process efficiency in our context and serves as a basis for the approach for identifying inefficiency drivers developed in this paper.

3.3 Related Work on Identifying Drivers of Inefficiency

Methods for identifying drivers of inefficiency are rare to find in academic literature. In the field of frontier analysis different approaches have been developed so far in order to analyze the determinants of inefficiency for lower ranked DMUs in a Data Envelopment Analysis. Most common are the so called *one-step approaches* and *two-step approaches*. In one-step approaches categorical variables are included in the DEA directly in order to account for environmental factors influencing the efficiency of DMUs. Within these approaches the production frontier is adjusted according to the group of DMUs being faced with certain common environmental factors, so that, as a consequence, the benchmarking occurs only towards the best fitting production function [16]. In contrast, two-step approaches include additional statistical tests for measuring the impact of specific variables on the efficiency scores. These approaches try to account for determinants of efficiency by applying a DEA analysis in the first place and to include a set of uncontrollable factors as independent variables in a tobit regression in the second step in order to seek to explain the variation in efficiency scores obtained by DEA [17; 18].

What these approaches have all in common is that they require a priori assumptions about what factors can possibly have an influence on the efficiency score. In one-step approaches this means defining the set of environmental factors to be included in the DEA, whereas in the two-step approaches this indicates selecting the appropriate independent variables for subsequent regression analysis. Hence, results of these approaches might fall short in identifying the real drivers of (in-)efficiency. The approach we present will not require these kinds of assumptions.

3.4 Research Gap and Implications for Research Design

Following previous findings summarized above, it can be stated that there is still research required in the field of developing an appropriate methodology for measuring process efficiency and subsequently identifying the drivers of inefficiency in

business processes. Our paper presents a methodology which enables to address the research question and the identified research gap. This methodology can further serve as the basis to build a workflow management add-on for process efficiency analysis. This paper has to be classified into Design Research. In particular, this paper will focus on the aspects of theory building in terms of *Develop/Build* according to [19]. Referring to [20], this part of research is the important step including the abstract activity of conceptualization, which they call *design*, and the physical realization of the designed artifact, which they call *build*. What we develop in this paper deals with the step of *develop* [19] or *design* [20] by providing the conceptualization of an artifact. As in the paper we describe the methods to be used, the rules how to link these different methods, and the expected outputs to be generated, we call our developed artifact a methodology, being in line with [21]. According to [19], a relevant artifact has to be further built upon *Business Needs* and *Applicable Knowledge*. The business need of our methodology is founded in the lack of appropriate tools for efficiency measurement and analysis for the specific perspective; the applicable knowledge is based on well-established methods from scientific research, i.e. DEA, cluster analysis, and association analysis. The physical realization will take place as the utilization or the adjustment of DEA- and Data Mining-software programs, which is able to follow the rules of the conceptualized methodology. The final evaluation can be conducted as a case study incorporating the application of the developed artifact.

4 Methodology for Identifying Drivers of Process Inefficiency

Our approach contains a combination of methods for identifying systematic drivers of inefficiency. It further overcomes the problem of deriving hypotheses on potential drivers of inefficiency beforehand. Instead, it focuses on identifying structural patterns in a large dataset. In the discipline of *data mining* many different methods for this purpose exist. Generally, “data mining refers to extracting or “mining“ knowledge from large amounts of data“ [22, p.5]. In this context it is critical to decide when an amount of data is considered to be “large” compared to be “small”. Previous approaches on measuring process efficiency with DEA compared processes with each other. In such a case the amount of DMUs is quite low. The approach of [7], however, employs process transactions as DMUs. A typical back office process, like e.g. the securities settlement and clearing, easily encounters more that 100,000 transactions per day at a large bank. As a consequence, measuring the intrinsic process efficiency by benchmarking of transactions leads to a dataset of efficiency scores that can be considered to be “large”. This opens up opportunities to further examine drivers of inefficiency by utilization of data mining methods. The objective is to identify structural associations of transaction characteristics and their efficiency scores. The methodology is comprised of three steps described in the following sections.

4.1 Step 1: Measuring Process Efficiency with Data Envelopment Analysis

The DEA efficiency scores are calculated for each (comparable) process transaction, following the approach of [7]. It should be noted, that DEA requires the DMUs to be comparable in order to meet the requirement of homogeneity [23]. To fulfill this

requirement, the sample of transactions to be chosen should encounter the same security type, time horizon, and processing route. Business processes are generally evaluated with regard to cycle time, costs, and quality. For the perspective of efficiency of standardized processes, a minimization of input utilization is required. Therefore, our analysis focuses on reducing inputs required for generating a certain output. As a consequence, an input-oriented CCR-model is applied in order to assess the intrinsic efficiency of the process [11]. The choice of inputs should always depend on the context of analysis. Since time and costs are the main drivers of process performance, it is vital to include them as inputs. Although back office operations of banks are rather automated, they require manual intervention for specific process activities or exception handling. These interventions are cost drivers having a high impact on the total process costs. As a consequence, including manual and automated processing time as two different inputs in the DEA seems to be appropriate proxies for process time and costs.

In order to ensure a purely input-oriented perspective on process performance, the outputs are normalized following the proposition of [24] and are set to the integer 1. Denoted in its dual model (envelopment form), for each transaction (DMU) i the following linear program is applied:

$$\begin{aligned} &\min_{\theta,\lambda} \quad \theta \\ \text{subject to} \quad &\theta x_i - X\lambda \geq 0 \\ &Y\lambda \geq y_i \\ &\lambda \geq 0 \end{aligned} \tag{1}$$

where θ is a scalar, λ is a non-negative vector, X is the vector of inputs (manual and automatic processing time) and Y the vector of outputs (integer values of 1). The result of step 1 is a DEA efficiency score between 0 and 1 for each transaction (DMU). An efficiency score of 1.0 indicates that the transaction is fully efficient, whereas a score of 0.8 indicates that there is an inefficiency of 20% in comparison to best practice, i.e. the reference transaction on the best practice frontier.

4.2 Step 2: Clustering Types of Efficiencies

Within the second step the DEA efficiency scores are grouped into categories of efficiency by applying a cluster analysis. The goal of the cluster analysis is to form groups of transactions which are similar in their efficiency scores and values of input variables. A partitioning method for cluster analysis is applied. Such a method is appropriate in cases of non-complex shapes of clusters [22]. Following the k-means method, the clusters are represented by the mean value of the cluster objects. The objects are assigned to clusters according to their nearest distance to the cluster mean. An iterative relocation is conducted in order to find the optimal allocation.

The results of step 1 are the values of both input variables and the calculated DEA efficiency score for each transaction. For the purpose of clustering, an inefficiency value is determined for each transaction by the following simple calculation.

$$IV_i = 1 - Eff_i \quad (2)$$

where IV_i is the inefficiency value for each DMU *i* and Eff_i is the DEA efficiency score calculated for each DMU *i*. The reason to transform the DEA efficiency score in an inefficiency value is that generally a high DEA score indicates high efficiency, whereas a high input value should be interpreted as a reason for *in*efficiency. Hence, the inefficiency value is calculated to simplify the interpretation of the results achieved with the cluster analysis.

The goal of step 2 is to identify clusters of transactions which are similar in the values of inputs and the inefficiency value. Since the DEA efficiency score has been determined by the value of input variables, there is a dependency between the inefficiency value and the input values. Thus, it can be expected that clusters similar to those summarized in Table 1 will be generated. The result of step 2 will be a DEA inefficiency value for each transaction and its allocation to a specific cluster.

Table 1. Examples of possible clusters identified in step 2

Cluster	Inefficiency Value	Manual Time	Auto Time
C1	High	High	High
C2	Average	High	Low
C3	Average	Low	High
C4	Low	Low	Low
…	…	…	…

4.3 Step 3: Deriving Association Rules

The aim of the cluster analysis in the previous step was to form groups of similarities concerning their input values and the corresponding inefficiency value. The next step is to identify causes for the inefficiency values. Here, an association analysis is applied in order to analyze which transaction characteristics have been most commonly associated with high, low or moderate inefficiency value. It now becomes apparent why the input values have been taken into account in the cluster analysis of step 2. An average inefficiency value can be either caused by high manual processing time and low automatic processing time or vice versa. The goal can be now, e.g. to identify causes which are associated with a high manual processing time and a high inefficiency value. The causes associated with high automatic processing time might be very different from causes associated with a high manual processing time.

An association analysis is a concept from data mining related to frequent pattern mining. It "searches for recurring relationships in a given data" [22, p. 227]. The aim of the association analysis in this case is to identify qualitative transaction characteristics which are found to be most frequently associated with the allocation of that transaction to a specific cluster. As a prerequisite for applying an association analysis, the transactions characteristics have to be transformed into categorical variables.

Before association rules can be derived, the database is analyzed for frequent itemsets [22]. Consider a set of items $I = \{I_1, I_2, I_3, \ldots, I_m; C_1, C_2, \ldots, C_p\}$ representing all

categories of transaction characteristics defined in the previous step (including the clusters as additional characteristics), and $D = \{t_1, t_2, t_3, \ldots, t_i\}$ is the set of all transactions of the transactional dataset (Table 2).

Table 2. Example for a transactional dataset

	Transaction ID	Cluster/ Category IDs
t_1	#1000	C1, I1, I2, I6, I8
t_2	#1001	C1, I2, I3, I5
t_3	#1002	C2, I4, I6
t_4	#1003	C3, I1, I2, I6
t_5	…	…

Now frequent itemsets of I in D have to be identified. The proposition here is to use the "Apriori algorithm" [22]. The procedure of the Apriori algorithm is shown in Figure 2.

The Apriori algorithm starts with scanning D for the frequency of the items in I. According to a predetermined threshold it is defined how often an itemset has to be counted in order to be a frequent itemset. This is called the *minimum support count*. After having counted the frequency of items in the database, a comparison which items' frequencies are above this threshold level occurs. It is continued only with the frequent items. The next step is to generate all possible dual combinations of items. Then again the database is scanned for the frequency of occurrence of these combinations of itemsets. Again, it is only continued with the itemsets that fulfill the minimum support count and all possible combinations of three items are checked for their frequency in the dataset. This algorithm continues until there are no more combinations of items supporting the minimum count threshold.

Each transaction consists of a transaction ID and a t-itemset T_i including its individual transaction characteristics and the cluster it has been assigned to. Assume $X = \{X_1, X_2, X_3, \ldots, X_o\}$ is the set of frequent itemsets identified by the Apriori algorithm, whereas e.g. $X_1 = \{C_1, I_2, I_3\}$. Now, association rules can be derived. Since the aim is to identify the drivers of inefficiency, only association rules between clusters and categories need to be derived.

For example, for the frequent itemset $X_1 = \{C_1, I_2, I_3\}$ it is only interesting to derive the following association rules:

(1) $C_1 \rightarrow I_2$ *Cluster 1 is associated with Item 2*
(2) $C_1 \rightarrow I_2, I_3$ *Cluster 1 is associated with Item 2 and Item 3*
(3) $C_1 \rightarrow I_3$ *Cluster 1 is associated with Item 3*

The aim is to find *strong* association rules fulfilling *minimum support* and *minimum confidence* [22]. For instance, the support of association rule (2) is calculated as the ratio of transactions t in D containing X_1, where any t_i contains X_i if and only if $\{C_1, I_2, I_3\}$ is equal to or is part of T_i, i.e. the t_i-itemset. The threshold for fulfilling minimum support has to be defined from case to case. The confidence for the association

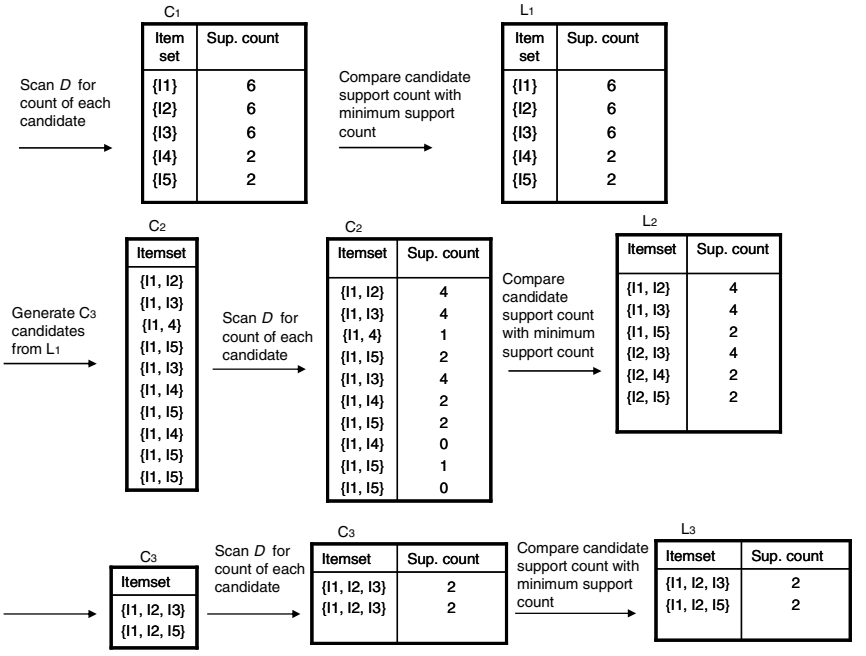


Fig. 2. The Apriori algorithm to identify frequent itemsets [22]

rule (2) is calculated as the support of $\{C_1, I_2, I_3\}$ divided by the support of {C1}. A confidence of e.g. 50% indicates that 50 percent of the transactions allocated to cluster one also exhibit the characteristics I_1 and I_2. Here, also a minimum threshold has to be defined depending on the context of the analysis. As a result, the identified strong association rules indicate which transaction characteristics are associated with specific clusters, i.e. efficiency categories.

This leaves room for interpreting these associations in terms of inefficiency drivers. The interpretation is one of the most important tasks now. Having generated the clusters according to the input variables and inefficiency values, this opens up the opportunity of more explicitly interpreting the transaction characteristics associated with e.g. manual processing time and a high inefficiency value. The interpretation of this case would be the drivers which frequently require manual intervention in the process, leading to an inefficiency of this transaction compared to the best practice transactions. Plausibility checks can make some of the associations found to be excluded and classified as coincidence. A further step is to decide how the identified inefficiency drivers have to be altered in order to lead to improvements in the process efficiency.

For additional analysis and robustness tests, measuring the statistical significance and magnitude of the inefficiency drivers on the process efficiency a regression analysis can be performed. Here, the dependent variable contains the DEA inefficiency values. The independent variables are those categories of transaction characteristics for which strong association rules have been identified.

5 Conclusion and Outlook

This paper provides the conceptualization of a three-stage approach for identifying the drivers of inefficiency in a business process. The measurement of efficiency from a business process perspective enables the assessment of banking production. Previous approaches to measure banking efficiency were tackled from a rather broad perspective, mainly on an organizational level. Focusing on banking production efficiencies however enables a more detailed detection of inefficiency drivers and, in turn, allows the derivation for more concrete action to achieve efficiency enhancements.

Our approach for detecting drivers of inefficiency in banking production functions combines DEA, cluster analysis, and association analysis. Transactions of business processes exhibit many different characteristics. However, in the back office operations of banks many transactions are processed that are similar in their characteristics. In addition, transactions and their characteristics can be unambiguously identified over the whole process flow as long as a workflow management system is available. As a consequence, identifying these characteristics, which are strongly associated with the inefficiency of many transactions, should lead to insights into the determinants of the process inefficiency. The method suggested in this paper allows to analyze a large amount of data and transaction characteristics and can further be the foundation for developing a workflow management system add-on for rather automating the detection of drivers of process inefficiency. The major advantage is that it requires no a priori assumptions about potential inefficiency drivers or the formulation of appropriate hypotheses. Thus, it allows the detection of inefficiency drivers there are not directly known by management beforehand.

The methodology presented in this paper mainly adds to current research as follows: (a) It provides an approach for process efficiency measurement and subsequent identification of inefficiency drivers in the process execution, (b) the methodology is based on methods which are more sophisticated than currently applied metrics for measuring process efficiency in practice, (c) the methodology does not require a priori assumptions about potential inefficiency drivers, but instead focuses on structural patterns in a large dataset of transactions, and (d) it can serve as the foundation for developing a workflow management system add-on, enabling automated identification of process inefficiency drivers.

The research activities concerning this project are to be continued. First, it has to be elaborated in more detail, if the k-mean method is the most appropriate type of cluster analysis. In recent research, neural network based cluster analyses are becoming more and more prevalent. For example, a Self Organizing Map (SOM) is especially useful in cases where high-dimensional data has to be clustered [22]. This means, if DEA is based on many input and output variables, it might be more appropriate to use a SOM clustering. Second, in order to show the applicability of this approach in terms of *Evaluate/Justify* proposed by [19], a case study will be performed for a relevant back office process of one of Europe's largest banks. The results of this case study will give more insights into the applicability and the strengths and weaknesses of the approach presented in the paper. However, the research results attained so far indicate that the presented three-stage approach for identifying drivers on inefficiency is a worthwhile topic to be further examined in future research.

References

1. Davies, M.N.: Bank-office process management in the Financial Services: A Simulation Approach Using a Model Generator. JORS 45(12), 1363–1373 (1994)
2. Cantner, U., Krüger, J., Hanusch, H.: Produktivitäts- und Effizienzanalyse. Der nichtparametrische Ansatz. Springer, Heidelberg (2007)
3. Thanassoulis, E.: Introduction to the Theory and Application of Data Envelopment Analysis. A foundation text with integrated software. Springer, New York (2001)
4. Cinca, C.S., Molinero, C.M., García, F.C.: Behind DEA Efficiency in Financial Institutions. Discussion Papers in Accounting and Finance, University of Southampton (2002)
5. Schmiedel, H., Malkamäki, M., Tarkka, J.: Economies of scale and technological development in securities depository and settlement systems. JBF 30(6), 1738–1806 (2006)
6. Frei, F.X., Harker, P.T.: Measuring the Efficiency of Service Delivery Processes: An Application to Retail Banking. JSR 1(4), 300–312 (1999)
7. Burger, A.: Analyse der intrinsischen Effizienz auf Prozessebene. Benchmarking von Transaktionen am Beispiel eines bankbetrieblichen Prozesses. In: Moormann, J. (ed.) Advances in Business Process Management. Logos, Berlin (2009)
8. Neely, A., Richards, H., Mills, J., Platts, K., Bourne, M.: Designing performance measures: a structured approach. Int. J. Oper. Prod. Man. 17(11), 1131–1152 (1997)
9. Valiris, G., Glykas, M.: Critical review of existing BPR methodologies: The need for a holistic approach. BPMJ 5(1), 65–86 (1999)
10. Berger, A.N., Humphrey, D.B.: Efficiency of Financial Institutions: International Survey and Directions for Future Research. EJOR 98(2), 175–212 (1997)
11. Charnes, A., Cooper, W.W., Rhodes, E.: Measuring the efficiency of decision making units. EJOR 2(6), 429–444 (1978)
12. Cooper, W.W., Seiford, L.M., Zhu, J.: Data Envelopment Analysis: History, Models and Interpretations. In: Cooper, W.W., Seiford, L.M., Zhu, J. (eds.) Handbook on Data Envelopment Analysis, pp. 1–39. Kluwer Academic, Boston (2004)
13. Soteriou, A., Zenios, S.A.: Operations, Quality, and Profitability in the Provision of Banking Services. ManSci. 45(9), 1221–1238 (1999)
14. Seol, H., Choi, J., Park, G., Park, Y.: A framework for benchmarking service process using data envelopment analysis and decision tree. ESWA 32(2), 432–440 (2007)
15. Kueng, P., Meier, A., Wettstein, T.: Performance Measurement Systems must be engineered. CAIS 7(1), 1–27 (2001)
16. Banker, R.D., Morey, R.C.: The use of Categorical Variables in Data Envelopment Analysis. ManSci. 32(12), 1613–1627 (1986)
17. Byrnes, P., Färe, R., Grosskopf, S., Knox Lovell, C.A.: The Effect of Unions on Productivity: U.S. Surface Mining of Coal. ManSci. 34(9), 1037–1053 (1988)
18. Sexton, T.R., Leiken, A.M., Nolan, M.S., Less, S., Hogan, A., Silkman, R.H.: Evaluating Managerial Efficiency of Veterans Administration Medical Centers Using Data Envelopment Analysis. MEDICARE 27(12), 1175–1188 (1989)
19. Hevner, A.R., Ram, S., March, S.T., Park, J.: Design Science in Information Systems Research. MISQ 28(1), 75–106 (2004)
20. March, S., Smith, G.: Design and Natural Science Research on Information Technology. DSS 15(4), 251–266 (1995)

21. Becker, J., Knackstedt, R., Holten, R., Hansmann, H., Neumann, S.: Konstruktion von Methodiken: Vorschläge für eine begriffliche Grundlegung und domänenspezifische Anwendungsbeispiele, Working Paper No. 77, Instituts für Wirtschaftsinformatik Münster (2001)
22. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, Amsterdam (2006)
23. Golany, B., Roll, Y.: An Application Procedure for DEA. Omega 17(3), 237–250 (1989)
24. Dyson, R.G., Allen, R., Camaho, S., Podinovski, C.S., Sarrico, C.S., Shale, E.A.: Pitfalls and Protocols in DEA. EJOR 132(2), 245–259 (2001)

An Enterprise Architecture Framework for Integrating the Multiple Perspectives of Business Processes

Eng Chew and Michael Soanes

Department of Information Technology, University of Technology, Sydney, Australia
{Eng.Chew,Michael.G.Soanes}@uts.edu.au

Abstract. Existing business process design strategies do not address the full breadth and depth characteristics of business processes. Multiple perspectives of business process design must be supported and integrated. Enterprise architecture frameworks provide a useful context to define and categorise these multiple perspectives. Levels of abstraction of business, systems and technology represent the lifecycle phase ranging from business requirements definition through to execution. Different deliverables are relevant to each level of abstraction. The business architecture consists of a set of modeling perspectives (process, activity, resource and management) that represent types of business requirements. The technology architecture defines a classification of execution architectural styles. The systems architecture consists of a meta-model that defines the fundamental concepts underlying business requirements definition facilitating the integration of multiple modeling perspectives and mapping to multiple execution architectural styles, thereby facilitating execution of the business requirements.

Keywords: enterprise architecture framework, business process modeling perspectives.

1 Introduction

There have been multiple similar proposals (Harmon [1], Davenport [2], Soanes[3]) advocating that existing business process design strategies do not adequately support the full range of characteristics of business processes. Soanes introduced the concept of the breadth / depth complexity matrix to define the characteristics of business processes. The matrix is defined as the combination of breadth (the range of activity types from structured to ad-hoc) and depth (the range of abstraction levels from coarse to fine grained) resulting in four quadrants (structured/coarse, ad-hoc/coarse etc) as per figure 1. Soanes also proposed that a typical business process has a breadth / depth coverage that crosses multiple breadth / depth matrix quadrants. Existing business process design strategies (and their associated modeling toolsets) tend to specialise in one quadrant, resulting in incomplete (that is, parts of the process design are not formalised), or at best, fractured process designs. Un-formalised parts of the process design, results in hidden business activity that cannot be planned and controlled and more importantly, cannot be monitored, thereby reducing operational transparency and accountability.

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 133–144, 2010.

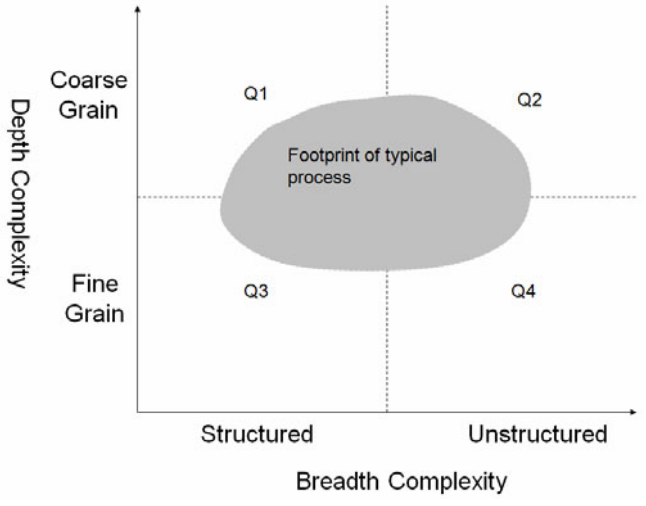


Fig. 1. Breadth / Depth Complexity Matrix

There are many stakeholders involved in business processes. However, the primary stakeholders addressed in this paper are customer, worker, resource manager (both internal and external suppliers) and business owner. Each stakeholder has a variable breadth/depth scope depending on their viewpoint (a worker has a more detailed viewpoint than a customer for example). Thus the need to support multiple overlapping business process definitions reflecting the breadth/depth scope appropriate for each stakeholder's viewpoint.

Reijers et al [4] define three dimensions (lifecycle phase, starting point and scope of improvement) to the evolution of business process designs as part of their proposed "process compass". This evolutionary view of business process design reinforces the need to support the evolution of breadth / depth scope within an ongoing iterative lifecycle of design through to implementation, with evaluation feedback from implementation to design.

Chew[5] has proposed that multiple perspectives of business processes have to be supported and integrated to address the breadth and depth characteristics of business processes. Existing process design strategies are appropriate for the scope of breadth / depth complexity they specialise in. The challenge is to identify the common underlying perspectives that facilitate integration across these multiple process design strategies, thereby supporting multiple stakeholder views of breadth / depth complexity.

The Open Group Architecture Framework (TOGAF) [6] proposes that an enterprise architecture framework (EAF) defines the dimensions and their associated deliverables as a common template for use within an architecture development method, to generate an organisation specific enterprise architecture. This paper will propose EAF as an appropriate artifact to define and categorise the multiple perspectives of business process design that are integrated and mapped across the lifecycle phases, thereby facilitating multiple stakeholder views of breadth/depth complexity. The objectives and design of meta-models as a key deliverable within the EAF will be introduced as the means of supporting the integration/mapping process. The use of

EAF's and meta-models to manage breadth/depth complexity is the key contribution of this paper.

2 Enterprise Architecture Framework Dimensions

There is no consistency across the various existing EAF's as to what are the dimensions and their associated deliverables that need to be supported. The Zachman framework [7] defines six levels of abstraction (representing the different stakeholders e.g. developer, builder etc) and six different modeling perspectives (e.g. function, data etc). The Object Management Group's (OMG's) Model Driven Architecture (MDA) [8] defines three levels of abstraction: computation independent model, platform independent model and platform specific model. TOGAF defines three levels of abstraction: business architecture, systems architecture and technology architecture. For both TOGAF and MDA, the modeling perspectives are represented by the deliverables proposed for each abstraction level. For example TOGAF defines the deliverables of the business architecture as business strategy, governance, organisation and business processes.

The Zachman framework does not attempt to integrate deliverables; a modeling perspective maps one to one through each level of abstraction. This will not satisfy the objective of integration of modeling perspectives described in the previous section. MDA has an objective of interoperability between multiple modeling notations and automating the mapping between levels of abstractions. However its scope is highly structured business domains and as such does not support the definition and integration of multiple modeling perspectives. TOGAF has recently released (Version 9.0 February 2009[9]) a meta-model to support the linking of deliverables across the levels of abstraction. The emphasis is on "linking" as distinct from "integration."

Consequently, it is proposed that these existing EAF's will not support the integration objectives defined in the previous section. It is proposed to combine terminology and concepts from existing frameworks extended with further refinements to address deficiencies identified with the existing frameworks to achieve the desired integration objective as per figure 2 below. As per Zachman and OMG's terminology, the term "levels of abstraction" defines the lifecycle phase ranging from defining business requirements through to execution. TOGAF's specific layers of business, systems and technology have been adopted as the levels of abstraction as defined in figure 2. This choice is based upon the reduced granularity of abstractions (three vs. six in Zachman) and their definitions are more aligned with the proposed deliverables (although OMG's definition of its three levels of abstraction is very similar to TOGAF's and could be adopted as an alternative terminology and definition). The term "deliverables" defines the artifacts that are produced at each level of abstraction. The term "modeling perspectives" specifically relates to a key deliverable within the business architecture that defines the different types of business requirements that need to be modeled (as per Zachman's terminology and concept). Finally the term "multiple dimensions" is proposed to be more applicable as a broader concept of the combination of levels of abstractions and the deliverables of each abstraction level (in effect the total set as defined in Figure 2). The term "stakeholder viewpoints" is proposed to represent a specific stakeholder's scope of interest of the total multiple dimension set. In Zachman, each level of abstraction is a stakeholder (eg system logic as a level of abstraction is

Level of Abstraction		Deliverables for each Level of Abstraction
Business Architecture	Definition of the business including strategy, governance, structure and processes.	Modeling perspectives
		Modeling constructs
		Modeling notations / languages
Systems Architecture	Structure of the components, their relationships and the principles of their design and evolution over time.	Mapping to Business Architecture
		Meta-model
		Mapping to Systems Architecture
Technology Architecture	The software and hardware infrastructure required to support the deployment of systems components	Execution architectural styles
		Specific execution standards and / or specific vendor products

Fig. 2. Proposed enterprise architecture framework

assigned to architects and designers). However it is proposed that it is more useful to allow stakeholders to have a scope of interest across multiple levels of abstraction and thus the need to support a separate concept of stakeholder viewpoints.

The following sections will first introduce the role and objectives of a meta-model in supporting the integration of multiple modeling perspectives and mapping them across the multiple levels of abstraction. Further detailed explanation of the deliverables in each level of abstraction is then provided.

3 Modeling Perspectives Integration and Mapping

Generally a meta-model is a collection of concepts (constructs, rules, terms) needed to build specific models (which are instantiations of the generic meta-model) within a domain of interest. Specifically within the scope of the proposed EAF, a meta-model is a definition of the fundamental concepts underlying business requirements definition facilitating the integration of multiple modeling perspectives (and their multiple modeling constructs/notations) and mapping to multiple execution architectural styles, thereby facilitating execution of the business requirements.

The Zachman framework provides little support for integration/mapping. TOGAF originally drove integration/mapping through a structured methodology rather than through integrating the deliverables of each perspective. TOGAF Version 9 [9] launched in 2009 has added a meta-model that links the artifacts together. OMG [8] have released a meta-model called Business Process Definition Meta-model (BPDM) [10] that aims to support the mapping of multiple procedural modeling notations to a Service Oriented Architecture (SOA) execution architectural style. BPDM remains

within a very structured prescriptive approach and would not address the breadth / depth coverage issue. The Super project [11] is a European Union project to implement semantic web concepts within the business process management (BPM) domain with the objective of raising BPM from the IT level to the business level. A specific goal is to integrate procedural and declarative modeling constructs through a set of meta-models (they call them ontologies). Although the most promising from an integration objective, it would still remain a very process centric approach.

Extrapolating from the proposed EAF in figure 2 and the role of the meta-model within the EAF, the following is a proposed set of six objectives of the meta-model. (1) To support the mapping of multiple modeling perspectives of the business architecture (in particular modeling constructs/notations combinations within each modeling perspective) to the meta-model facilitating the integration and conflict resolution of business requirements defined in each modeling perspective. (2) To support interoperability between modeling constructs/notations allowing the reverse mapping back from the meta-model to any modeling construct/notation retaining where appropriate the presentation semantics used in the original requirements definition construct/notation. (3)The reverse mapping from the meta-model to a modeling construct/modeling notation for presentation, should support the filtering of the business requirements to multiple levels of presentation detail reflecting the intended audience (e.g. a worker would require presentation of detailed business requirements while a manager would require presentation of high level business requirements). (4) The mapping from the meta-model to multiple execution architectural styles within the technology architecture to facilitate execution of the business requirements. (5) To support an evolution of design from execution feedback through an iterative design/execute/evaluate lifecycle. (6) The mappings to be automatable as much as possible with required manual intervention clearly defined.

4 Business Architecture

The business architecture defines the structure and operations of a business (where business processes is a subset). Within the business architecture there are three sub layers. The modeling perspective defines the type of business requirements that need to be modeled. Process, activity, resource and management are the proposed modeling perspectives as described in more detail below. Modeling constructs are conceptual approaches to modeling each business requirements perspective. For example, business processes (as a modeling perspective) can be modeled as prescriptive activity flows (as a modeling construct appropriate for highly structured aspects of processes) or declaratively as business rules (as a modeling construct appropriate for unstructured processes and exception handling aspects). Modeling notations / languages are the specific standards or vendor products that are used as implementations of different modeling constructs. For example, for prescriptive activity flow modeling as a modeling construct, there are multiple graphical notations including Business Process Modeling Notation (BPMN) [12] and Unified Modeling Language (UML) Activity Diagrams [13]. Modeling constructs can have multiple levels of abstraction as alternative approaches to addressing each modeling perspective. In the case of prescriptive activity flow: use case scenarios is a high level construct (where there are standard notations such as proposed

within UML); graphical activity flow modeling as described above is a mid level construct and procedural code is a low level construct (which has a multitude of language options).

The business architecture needs to support the modeling of multiple perspectives of the business. As described in section 2 above, existing EAF's are not consistent with their choice of modeling perspectives. Chew [5] introduced the Process / Activity / Resource / Management (PARM) framework as a model of business requirements perspectives that need to be supported and integrated to address breadth / depth complexity as follows:

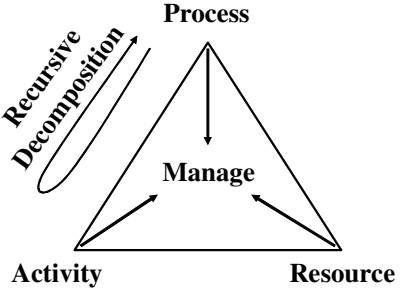


Fig. 3. PARM Framework – Modeling Perspectives

The process modeling perspective (MP) focuses on controlling, guiding and restricting the sequence of activities performed for specific process instances. Its measurable objective is to meet the customer's end to end service delivery expectations. The activity MP focuses on the facilitation of an environment to manage human activity with the recognition that human resources will prioritise their own execution of multiple activities across multiple processes simultaneously based upon their own individual work practices. Its measurable objective is to provide the most effective (both productivity and quality) environment for the completion of all work across all processes for the process or knowledge worker. The difference between a process and an activity is not definitive given that an activity can be a sub process decomposed at the next level of granularity. As a guideline rather than a rule, a process is sequencing non-contiguous activities over a long business lifecycle across multiple resources as compared to an activity, which is coordinating contiguous actions (sub-activities) within a system processing lifecycle usually by the one resource. The resource MP forecasts, plans, schedules and assigns resources to activities. Its measurable objective is to maximize the utilisation and therefore the efficiency of the total resource pool. This MP captures the resource manager's (e.g. team manager) requirements. The Management MP integrates the process, activity and resource MP's through balancing the tension between service, cost and quality expectations. It reflects the requirements of the business owner of the process.

An issue is whether PARM's choice of modeling perspectives to be integrated are complete and appropriate. In particular, social/people and information/data modeling perspectives are further candidates to be considered. As a first pass in the definition of the total strategy, the scope will be restricted to the four PARM perspectives.

5 Systems Architecture

A systems architecture defines the components of the system, their relationships and the guidelines and principles governing their design and evolution. The key deliverable within the system architecture is a meta-model which is the key enabler of modeling perspective integration and mapping across levels of abstraction as per figure 4 below.

The meta-model is positioned within a Service Component Architecture (SCA) [14] focus (which is a current in progress standard proposed as an evolution to Service Oriented Architecture (SOA)) and thus the adoption of terminology of "components" and "services" as proposed within SCA. Abstracting and encapsulating the business domain into components at multiple nested levels, facilitates management of depth complexity. Meta-model processing applies to the highest level component model and privately within each component at each nested level.

In essence, the fundamental concept of performing an individual instance of work, is the allocation of a resource to an activity that is to be performed on the process instance based upon the current state of the process instance, resulting in the transition to a new state, thereby triggering the need for further work appropriate for the new state. This cycle continues until no further work is required on that process instance. Each individual work instance exists within a total set of work across multiple processes, activities and resources where the prioritisation of each work instance is driven by cost, quality and schedule drivers of the various stakeholders (customer, worker, resource manager, business owner).

There are three key business operational logic definitions inherent in the above work definition. Firstly, defining what are valid states and what are valid and invalid transitions between these states, including where known, what is the conditional routing logic to trigger each valid state transition. Conditional routing logic can be formally defined as part of the process definition but also maybe unknown as it is embedded as part of the activity processing logic or embedded as human resource expertise. The second business operational logic definition is what activity (or activities) are to be performed for a process instance in a given current state. This current state/activities to be performed association may be formally defined (particularly for automated system activities) or maybe unknown as it is part of a human resource's role of deciding what manual activities are appropriate to perform for the process instance current state. The third business operational logic definition is defining what resource is required to perform the activity (ies) for the process instance in its current state. This current state/resource association can be formally defined allowing work to be pushed to the appropriate resource, or undefined requiring resources to pull work of the appropriate state.

Extracting from the above fundamental concept of performing work, the meta-model proposes that the triple relationship of process instance state transition (hereafter shortened to state transition), activity performed and resource usage is the fundamental conceptual structure to be modeled. Ideally all three associations (state transition/activity/resource) would be defined, however as per the above description of work, state transition is the core and associations with activity and resource are optional. Reflecting the importance of state transitions, this view of work will be called the "state based view of work" (as distinct to a traditional activity flow based view of work).

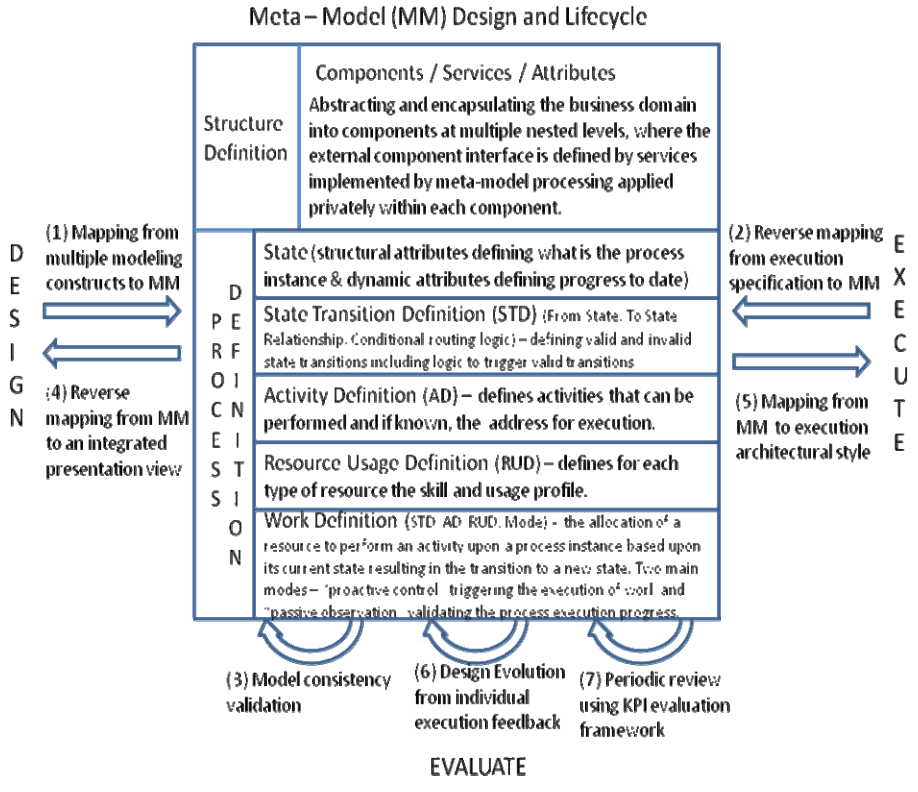


Fig. 4. Meta-Model Design and Lifecycle

A state can be any set of attributes and their values appropriate to the business domain. However there are structural and dynamic components to state definition. The structural component defines what the process instance is. A common set of attributes would be process type, client type and product type (although they would vary per business domain). They are structural in the sense that a process instance will not change these attributes frequently, if at all. The dynamic state definition component defines what has happened so far to the process instance. A common set of attributes would be business state and system state as a summary of the existing state, although detailed data attributes could be used.

There are three principles it is proposed that the work definition concept must support to assist the meta-model integration/mapping objectives. Firstly, the work definition concept through the "relationship" attribute will allow the definition of both definitive state transitions (transitions that either "must" happen, "should" happen or "can" happen) and restrictive state transitions (transitions that "should not" happen or "must not" happen). Traditional prescriptive process definition approaches do not support restrictive process definition although declarative rules type approaches would support both definitive and restrictive. Restrictive process definition is useful for defining compliance type business requirements. Secondly, the work definition

concept through the "mode" attribute will support a proactive control mode (where meta-model processing is the trigger for business process execution) and a passive observation mode (where business processing is occurring beyond the control of the meta-model). In the passive mode, the meta-model is either providing proxy services (creating defined activity events from known state transitions and vice versa) or reconciliation services (where state transition events are reconciled back to activity events and vice versa). Finally, a fundamental meta-model design feature is that the state transitions are implemented as a hierarchy of transitions where state transitions can overlap or be duplicated at multiple breadth and depth levels of the hierarchy. For example, take the simple state transition flow: A transists to B and B can transist to either C, D or E. Wild card character definition supports variable breadth definitions. For example, if B / C is the core transition and B/D and B/E are low volume exception cases, then an appropriate breadth definition could be B/C as an explicit definition and B/* to rollup all exception cases. Multiple / overlapping depth definitions would support detailed definition of A/B, B/C, B/* transitions as well as support higher level duplicated definitions of A/C and A/*. This multi-filtered hierarchical definition of state transitions is an important enabler of supporting varying breadth / depth scopes for the multiple modeling perspectives.

Having defined the design of the meta-model, the lifecycle of the meta-model will be explained. A state based view of work definition tends to be a more evolutionary definition lifecycle as represented in the iterative design / execute / evaluate lifecycle in figure 4. The ability to succinctly define in advance all possible valid state transitions is difficult and thus work definition must be defined within an ongoing feedback cycle adjusting appropriately with new or modified state transitions when execution feedback indicates that state transitions are occurring that had not been envisaged. It is proposed that this more accurately reflects the complex reality of business operations than traditional static activity flow business process design.

The mapping from multiple modeling constructs to the meta-model (lifecycle phase 1 in figure 4 above) represents the key phase of mapping from the business architecture to the system architecture. It would be an ideal capability to reverse map the processing definitions from existing legacy execution environments back into the meta-model (lifecycle phase 2). Having populated the meta-model from multiple modeling constructs and the reverse mapping from existing legacy environments, consistency validation logic (lifecycle phase 3) is necessary to analyse the meta-model to detect issues such as overlapping state transition definitions and inconsistent state transition definitions. Reverse mapping back from the meta-model to integrated modeling constructs (lifecycle phase 4) provides a consolidated view of the multiple perspectives at multiple levels of detail appropriate to the audience. After this consolidated review, the meta-model content can now be mapped to suitable execution architectures (lifecycle phase 5) representing the key phase of mapping from the system architecture to the technology architecture. As work is performed within the execution environments, event feedback will be received. There is scope to review and evolve the meta-model content based upon each work event (lifecycle phase 6). For example, where those events represent state transitions that have not been previously defined. Finally, there is an automated periodic review (lifecycle phase 7) of the existing state definitions and their breadth and depth granularity definition based upon aggregated statistical feedback of actual execution events. For example, very low frequency exception based

state transitions maybe rolled up into higher level summarised state transition definitions to avoid an unmanageable explosion in exception case definitions, whereas high volume state transitions justify detailed definition (although additionally influenced by stakeholder viewpoints). Supporting the evaluate processing is a key performance indicator (KPI) evaluation framework that measures the depth and breadth design effectiveness of the state based meta-model definition. The full definition of this KPI evaluation framework is the subject of a future paper.

6 Technology Architecture

The technology architecture defines the software and hardware infrastructure intended to support the deployment of system components. The technology architecture has two levels. Execution architectural styles define different categories of approaches to executing business requirements that in the second level may have multiple standards or vendor specific products that implement that execution style.

Many architectural styles have evolved as a means of supporting this deployment. A classification of architectural styles is proposed by Fielding [15]. Fielding bases his categorisation on the constraints inherent in the communication of components of the system. Fielding defines twenty-two styles on this basis with the recognition that there are further possible styles. Examples of Fielding's styles are client server, mobile agent, event based integration and distributed objects. It is out of scope of this paper to define in detail the multitude of architectural styles and standards in the implementation domain. The choice will be influenced by specific business domain requirements and the existing legacy implementation environment.

However, this paper proposes a simplified categorisation of architectural styles based upon process design coupling (how strong is the dependency between participants of the process) and process design cohesion (how centralised is the control of process flow) as proposed in the following diagram:

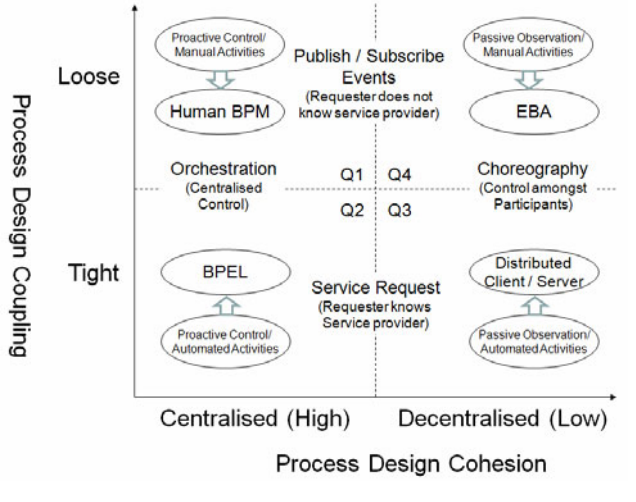


Fig. 5. Categorisation of execution architectural styles

As per the definition of the meta-model in section 5 above, it is proposed that there are two modes of meta-model operation: proactive control and passive observation. In applying these modes to an execution environment, an additional dimension of whether the activities are manual human resource executed or automated systems activities is useful to classify the appropriate execution architectural style for each mode. Thus we have four meta-model operational modes: proactive control/manual activities (meta-model facilitates process execution but requires a resource as link between process and activities); proactive control / automated activities (meta-model facilitates process execution and has direct control over activities); passive observation / automated activities (meta-model validates but does not control process execution within an environment of automated system activities) and passive observation / manual activities (meta-model validates but does not control process execution within an environment of human resources performing activities).

Although not definitive, the four meta-model operational modes map to the four quadrants of the simplified categorisation of execution architectural styles as per figure 5. Additionally, although not the only choice of example execution architectural styles for each quadrant, the following will propose example execution architectural styles that map to the four meta-model operational modes. Human based BPM is the typical solution used for highly centralised process control (through a workflow engine) with loose coupling, where the human work performers can be easily interchanged (as defined in quadrant 1). This is a suitable execution architectural style for the proactive/manual meta-model operational mode. Business Process Execution Language (BPEL) [16] has evolved as a common standard for supporting system based business process management (reflecting the history of BPEL as evolving from enterprise application integration (EAI) where message passing between applications is the key BPM requirement). BPEL is more appropriate for quadrant 2 reflecting tight coupling (client knows server) and highly centralised process coordination. This is a suitable execution architectural style for the proactive/automated meta-model operational mode. Tight coupling (client knows server) but distributed (and also evolutionary) control amongst participants as defined in quadrant 3 is typical of a distributed client server type architecture where the process evolves over the evolution of multiple interactions amongst clients and servers. This is a suitable execution architectural style for the passive/automated meta-model operational mode. Finally, the preferred architecture (as it is the most flexible) is an event based architecture (EBA) where control is decentralised amongst participants and where communication between participants occurs via events that are published and interested participants subscribe (as defined in quadrant 4). The mapping of example architectural styles to the specific quadrants does not negate that each style can additionally (but maybe not ideally) address meta-model operational modes mapped to the other quadrants. In particular, it is proposed that EBA is suitable for all meta-model operational modes and additionally is useful as a means of integrating process execution across multiple execution architectural styles.

7 Conclusion

EAF's provide a useful context to define and categorise the multiple perspectives of business processes necessary to support the full breadth and depth characteristics of

business processes. A specific enterprise architecture framework was proposed, that defined the levels of abstraction representing the lifecycle phase from business requirements to execution. Deliverables at each level of abstraction were defined. Specifically a set of modeling perspectives were proposed (representing the types of business requirements) for the business architecture. A meta-model was proposed for the systems architecture. A categorisation of execution architectural styles was proposed for the technology architecture. The use of EAF's and meta-models to manage breadth/depth complexity is the key contribution of this paper

It was beyond the scope of this paper to define the mappings between each level of abstraction which will be future work. The total solution will then be applied to common business process scenarios that have a mixture of breadth / depth complexity profiles using a case study as real examples of each scenario.

References

1. Harmon, P.: Task Complexity Continuum. BPTrends (July 18th, 2006) published at, http://www.bptrends.com
2. Davenport, T.H.: Thinking for a Living., p. 27. Harvard Business School Press, Boston (2005)
3. Soanes, M.G.: Process Design Strategies to Address Breadth and Depth Complexity. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102. Springer, Heidelberg (2006)
4. Reijers, H.A., Mansar, S.L., Rosemann, M.: From the Editors: Introduction and a Compass for Business Process Design. Information Systems Management 25, 299–301
5. Chew, E., Hawryszkiewycz, I., Soanes, M.: Value Configuration Design – an evolution in adequate business process design. In: 8th Workshop on Business Process Modeling, Development and Support at CAISE 2007, Trondheim Norway (June 2007)
6. TOGAF accessed at, http://www.opengroup.org/togaf
7. Zachman Framework accessed at, http://www.zachmaninternational.com
8. Model Driven Architecture accessed at, http://www.omg.org/mda
9. TOGAF Version 9 accessed at, http://www.opengroup.org/togaf
10. Business Process definition Meta-model accessed at, http://www.omg.org
11. Semantics Utilised for Process Management within and between Enterprises (SUPER) project accessed at, http://www.ip-super.org
12. Business Process Modeling Notation accessed at, http://www.bpmn.org
13. UML Activity diagrams as part of UML 2.0 at, http://www.uml.org
14. Service Component Architecture accessed at, http://www.osoa.org/
15. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine (2000)
16. Business Process Execution Language (BPEL) accessed at, http://www.oasis-open.org

An Interperspective-Oriented Business Process Modeling Approach

Marcel Fouda Ndjodo[1], Priso Essawe Ndedi[1], and Roger Atsa Etoundi[2]

[1] Department of Informatics and Educational Technology
ENS-University of Yaounde I, P.O. Box 47 Yaounde, Cameroon
{marcel.fouda,priso}@dite-ens.cm
http://www.dite-ens.cm
[2] Department of Computer Science
Faculty of Science-University of Yaounde I, P.O. Box 812 Yaounde, Cameroon
roger.atsa@uy1.uninet.cm

Abstract. In this paper, we propose a business process modeling approach based on the concept of process environment, by using a set of observers. The execution conditions of a task are linked to the environment rather than a predefined order between tasks. Relying on the environment and the tasks, a formal definition of a business process is given. The modeling proposal presented in this paper is used to address three non dominant perspectives: context, semantics and goal; along with the usual control-flow perspective.

Keywords: Business Process, Perspectives, Goal, Context, Semantics.

1 Introduction

Many techniques have been used to model business processes and workflows, from the very intuitive graph theory [22], to more sophisticated mathematical models such as Petri Nets[1] and, more recently $Pi - Calculus$ [20].

Regardless of the modeling capacities of the available modeling frameworks, in practice, the control-flow perspective seems to have become the dominant perspective [2]. While we admit that encapsulating all the perspectives in one model is not realistic, it has been proven that it is possible to build a model that natively considers more than one perspective without extra-complexity, as shown in [21] for the data perspective, or in [11] for the quality of service perspective .

This paper presents a modeling framework that is used to address three non dominant perspectives: context, semantics and goal; along with the usual control-flow perspective. In this model, business process environments, i.e. the execution contexts of business processes - defined here as a set of boolean observers - are modified by the tasks which have the capacity of changing the values of observers, thus modifying the states of the environments. The concept of environment used here is similar to the Leslie Lamport's environments [15]. The execution conditions of a task are linked to the environment rather than a predefined order

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 145–156, 2010.

between tasks. Relying on the environment and the tasks, a formal definition of a business process is given.

The core model supporting this approach is described in [5,6], and it is supported by a number of papers [7,8,9] that illustrate its multi-perspective nature, giving room for further investigations.

The rest of the paper is strutured as follows: section 2 presents some related works; the model is given in section 3 which is the main part of the work; section 4 illustrates the multiperspective nature of the model; and section 5 concludes the paper.

2 Related Works

This section roughly summarizes how some main modeling approaches deal with the multiperspective nature of business processes.

Graph theory have been widely used to model business processes. But, as it is expressed in [22] the initial weaknesses of this approach, such as the difficulties of reasoning about workflow properties, and the impossibility to express global dependencies have opened ways to enrich the initial model by adding constraints such as path constraint [22] and temporal constraints [10]. Despite all these improvements, it is hard to model other perspectives than the control-flow. This is probably one of the reasons which explains why this approach has not emerged as one of the most important for business process modeling.

The Workflow-nets approach [1], based on Petri Nets, models tasks by transitions, conditions by places, and cases by tokens. This approach mixes a formal semantics to the graphical and intuitive nature of Petri Nets [4]. Despite a number of extensions available for Petri nets such as coloured Petri nets, timed Petri nets and hierarchical Petri nets that suggest the handling of many perspectives, the full power of Workflow-nets (in terms of the abundance of analysis techniques and software tools made available for it by a very prolific research community) needs the abstraction from perspectives other than control-flow and data to be expressed [14].

However, in [21], workflow-nets are extended with data. Each task is associated with three sets that indicate which data elements the task reads, writes or destroys. This extension allows analysis techniques to check for errors such as deadlocks or livelocks without abstracting from the data perspective, and hence extending the detection power of such tools.

In [5,6], a modeling framework is presented that natively addresses the context, control-flow, time and resource perspectives of business processes. The multiperspective nature of that approach has been illustrated by works focusing on the human resources [9], mobility [8] and security [7] aspects of business processes. This work falls within this approach.

3 The Context-Based Modeling Framework

The modeling framework used in this paper has been described in [6]. The cornerstone of this approach is the concept of environment which formalizes the

execution contexts of business processes defined here as a set of boolean observers. It is worthy to note that the notion of context used here is different from the notion of context used in [18], where a context is an assignment of a status (wait or dead) to arcs linking two nodes of a process in order to manage the arrival of tokens. Here, a context is a "means to focus on aspects that are relevant in a particular situation while ignoring others" [19]. The relevant aspects are captured through the concept of observers. A business process environment is therefore the set of relevant observers required for the execution of that business process.

The mutations of business process environments are enforced through business process tasks which have the capacity of changing the values of observers, thus modifying the states of the environments. The business process tasks are ordered by a follow function in order to achieve an expected goal.

The rest of the section is organized as follows: section 3.1 defines the business process environments, business process tasks are defined in section 3.2, section 3.3 gives a formal model of business processes and, section 3.4 shows that this model captures some common routing constructions. In section 3.5, an example to illustrate the model is given.

3.1 Business Process Environments

The context of execution of a business process is an important modeling input that determines a number of actions. As it is not possible to capture the entire context, we restrict ourselves to the part of the real world that is of interest for a business process. We call it the environment. We define an environment as a set of different metrics whose value may change [6]. Every relevant characteristic of the real word is captured through boolean objects that we call observers.

Definition 1. ***Environments***

An environment ξ is a tuple $< \theta, S, val >$ where:

- θ is a non empty set whose elements are called **observers;**
- S is a non empty set whose elements are called **states** ($\theta \cap S = \emptyset$);
- $val : \theta \rightarrow (S \rightarrow Bool)$ is a function which describes the behaviour of observers in the different states. □

When the context is clear, we write $s(o)$ for $val(o)(s)$ with the intuitive meaning that $s(o)$ is the value of the observer o in the state s.

Given an environment ξ, an *observation* tells us if a condition over a set of observers is satisfied or not. An observation therefore has a positive part and a negative part. The positive part of an observation is the set of the observers whose value is expected to be $true$, while the negative part is the set of observers whose value is expected to be $false$.

Definition 2. *Observations*

Let $\xi =< \theta, S, val >$ be an environment, an observation on ξ is a couple $< P, M >$ where P and M are disjoint sets of observers of θ. □

The set of the observations on the environment ξ is denoted $\mathcal{O}_\xi$. When the context is clear, we write $\mathcal{O}$ for $\mathcal{O}_\xi$.

Definition 3. *Satisfaction of an observation*

Given an environment $\xi =< \theta, S, val >$, the satisfaction of an observation $obs =< P, M > \in \mathcal{O}$ in a state $s \in S$ is given by the function

$\Phi : S \times \mathcal{O} \rightarrow Bool$ defined by:
$\Phi(s, obs) = (\forall o \in P, s(o)) \wedge (\forall o \in M, \neg s(o))$
we write $s(obs)$ for $\Phi(s, obs)$. □

Definition 4. *Gap between states*

Given an environment $\xi =< \theta, S, val >$, two states $s_1, s_2 \in S$, the gap between s_1 and s_2 (denoted $s_1 \underline{\bullet} s_2$) is defined by:
$s_1 \underline{\bullet} s_2 = \{o \in \theta : s_1(o) \neq s_2(o)\}$. □

The gap between two states s_1 and s_2 is the set of the observers that have different values in s_1 and s_2.

3.2 Business Process Tasks

The concept of task is not quite new in the modeling of workflows. All the workflow modeling approaches consider this concept [13]. In [5], a task is defined as a state transition function $task : S \rightarrow S$. This definition includes multipurpose tasks - that can produce different effects according to the initial state, and single-purpose tasks which produce the same effect (post-condition), whatever the initial state may be. Intuitively, any multipurpose task can be viewed as a "combination" of single purpose tasks with appropriate branching. Therefore, single-purpose tasks behave like atomic tasks. In this paper, we restrict ourselves to atomic tasks.

Definition 5. ***Tasks***

Let $\xi =< \theta, S, val >$ be an environment, a task on ξ is a triple $< t, ec, action >$ where t is the identifier of the task, ec is an observation specifying its pre-condition, and $action$ is an observation specifying its post-condition. □

In the rest of the paper, the execution condition (resp. post-condition) of the task $< t, ec, action >$ is denoted $ec(t)$ (resp. $action(t)$);

In the same vein, $P(action(t))$ (resp. $M(action(t))$) is denoted $P(t)$ (resp. $M(t)$).

Definition 6. *Conflicting Tasks*

Let t_1, t_2 be two tasks on the environment $\xi =< \theta, S, val >$, we say that t_1 and t_2 are conflicting tasks if:

$(P(t_1) \cap M(t_2) \neq \emptyset) \vee (P(t_2) \cap M(t_1) \neq \emptyset)$, i.e there exist an observer on which t_1 and t_2 have opposite actions. □

This notion can easily be extended to a set of tasks as follows: ts is a non conflicting set of tasks if: $\forall \{t_1, t_2\} \subseteq ts$, t_1 and t_2 are not conflicting tasks.

3.3 Formal Model for Business Processes

A business process is defined as "a set of logically related tasks performed to achieve a defined business outcome" [12] which is the goal of the business process. According to the section 3.2, any business process with multipurpose tasks can be modeled as a business process with single-purpose tasks.

In this framework, business processes are therefore formalized as follows:

Definition 7. ***Business Processes***

Given an environment ξ, a Business Process is a tuple $BP =< \theta, T, f, g >$ where:
- θ is a set of observers over ξ,
- T is a set of tasks on ξ,
- $g \in \mathcal{O}$ is a distinguished observation called the goal of BP.
- $f : T \rightarrow 2^T$ is a function which, for every task, gives the names of the tasks that can be executed right after it. □

We observe that in this definition, the follow function can be cyclic, allowing this framework to model loops.

The execution model, (i.e. the operational semantics) of the business process is defined through the definition of the execution of a non conflicting set of tasks.

Definition 8. ***Execution of non conflicting tasks***

Let ξ be an environment, $BP =< \theta, T, f, g >$ a business process over ξ, ts a non conflicting set of tasks, and s a state of ξ.

The execution of the set of tasks ts in the state s moves the environment into the state s', and activates the set of taks ts'such that:

- $s' \underline{\bullet}\ s = (\underset{t \in ts}{\cup} P(t)) \cup (\underset{t \in ts}{\cup} M(t))$ (the gap between s and s' is the set of observers modified by tasks of ts)
- $ts' = \{t \in \underset{t \in ts}{\cup} f(t) : \Phi(s', P(ec(t))) \wedge \Phi(s', M(ec(t)))\}$ (the followers of the tasks of ts whose execution conditions are satisfied in s')

We write $exec(ts, s) = (ts', s')$. □

The intuitive idea is that all the tasks whose execution conditions are satisfied in a state are concurrently executed, unless they are conflicting.

Definition 9. ***Implementations of Business Processes***

Let ξ be an environment, $BP =< \theta, T, f, g >$ a business process over ξ. An implementation of BP is a list of couples $< (ts_0, s_0), ...(ts_n, s_n) >$ where:

- s_i is a state,
- ts_i is a non conflicting set of tasks,
- $(ts_{k+1}, s_{k+1}) = exec(s_k, s_k)$
- g is satisfied in s_n. □

The complete investigation of the operational semantics is not the main purpose of this paper. A substancial amount of this investigations can be found in [5].

3.4 Common Routing Constructions Modeling

A model pretending to describe a workflow process shall implement some basic routing constructions that will guide the flow of work. Along with the sequential execution which is part of our business process definition and implemented by the f (follow) function, it should be possible to address parallel execution, switching and synchronization.

This section shows that execution conditions and the follow function are very powerful routing tools that will enable one to route work in all cases.

Definition 10. *Sequential execution*

The routing "t_2 is executed after t_1" is formalized as $f(t_1) = \{t_2\}$. □

Definition 11. *Parallel execution*

The routing "t_1 can be executed in parallel with t_2 after the task t" is formalized as:

$$\begin{cases} ec(t_1) = ec(t_2) \\ \{t_1, t_2\} \in f(t) \\ t_1 \text{ and } t_2 \text{ are not conflicting} \end{cases}$$ □

Definition 12. *Switch*

The routing "after t, either $t_1$or t_2 will execute, not both" is formalized as:

$$\begin{cases} (P(ec(t_1)) \cap M(ec(t_2)) \neq \emptyset) \vee (M(ec(t_1)) \cap P(ec(t_2)) \neq \emptyset) \\ \{t_1, t_2\} \in f(t) \end{cases}$$

This formalization says that at most one of the tasks t_1 and t_2 can be executed after t because their execution conditions are conflictual. □

Definition 13. *Synchronization*

The routing "t will be executed only when both t_1 and t_2 have been executed" is formalized as:

$$\begin{cases} (1)(P(ec(t)) \subset (P(t_1) \cup P(t_2)) \wedge (P(ec(t)) \nsubseteq P(t_1)) \wedge (P(ec(t)) \nsubseteq P(t_2)) \\ (2)(M(ec(t)) \subset (M(t_1) \cup M(t_2)) \wedge (M(ec(t)) \nsubseteq M(t_1)) \wedge (M(ec(t)) \nsubseteq M(t_2)) \end{cases}$$

whose meaning is that t can only be executed if both $t_1$and t_2 have been executed. None of them alone suffices for t to be executed. □

3.5 Example

In this section, let us use an example of business process found in [21] to illustrate the present model. The business process in [21] is described using well-known Petri nets. We slightly modified the original business process in order to better show the capabilities of this approach (see Fig. 1).

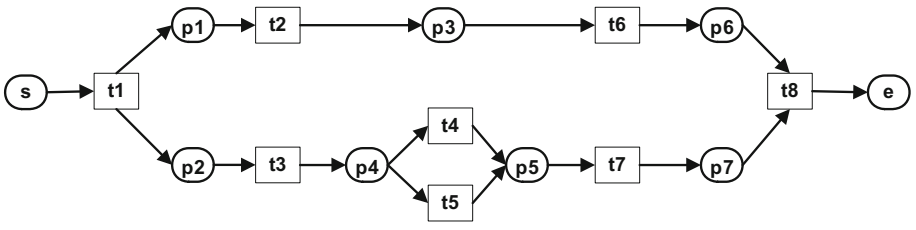


Fig. 1. Sample Business Process using Petri nets

The environment is made of 11 observers: $\theta = \{a, b, c, d, e, f, g, h, o, u, v\}$, We have 8 tasks to modify the state of the environment:$T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\}$. The business goal is satisfied if the observers u and v are set to $true$, and the observer o is set to $false$. Formally, the goal g is such that: $P(g) = \{u, v\}$ and $M(g) = \{o\}$. We now define how the tasks behave, and how they are routed.

Tasks	Pre	Post	Follow
t_1		$\{c, e, f\}$	$\{t_2, t_3\}$
t_2	$\{a\}$	$\{a, d, h\}$	$\{t_6\}$
t_3	$\{\neg a\}$	$\{b, v\}$	$\{t_4, t_5\}$
t_4	$\{b\}$	$\{g\}$	$\{t_7\}$
t_5	$\{b\}$	$\{u\}$	$\{t_7\}$
t_6	$\{a, h\}$	$\{u, v\}$	$\{t_8\}$
t_7	$\{g, u\}$	$\{f, h\}$	$\{t_8\}$
t_8	$\{f, h, u\}$	$\{\neg o\}$	$\varnothing$

The transition after t_1 is a switch, as the pre-conditions of t_2 and t_3 are incompatible (a for t_2 and $\neg a$ for t_3). So, according to the fact that t_1 does not modify the observer a, the choice between t_2 and t_3 will be determined by the initial state.

t_4 and t_5 have the same pre-condition (b), and they both belong to $f(t_3)$, so they can be executed in parallel. The pre-condition of t_7 is garanteed only if t_4 (for g) and t_5 (for u) are executed, so t_4 and t_5 synchronize to t_7.

Given a state $s_0 = \{a, d, e, g, h, o, v, \neg b, \neg c, \neg f, \neg u\}$ where the observers a, d, e, g, h, o, v have the value $true$, and the observers b, c, f, u have the value $false$, and given the input $< (\{t_1\}, s_0) >$, the tasks t_1,t_2, t_6 and t_8 will be executed, and the final state will be $s_f = \{a, c, d, e, f, g, h, u, v, \neg b, \neg o\}$ and the goal g will be satisfied as the observers u and v are $true$ and the observer o is false in the final state s_f.

4 Business Process Perspectives

The definition of a business process as a set of logically related tasks, directly suggests the control flow perspective of a business process. A coherent theory about business processes cannot ignore this perpective. But this perspective should not over-shadow the others whose importance have been stressed a long time ago [3].

In this section, we show how this model captures three usually uncovered perspectives. The first one is the context perpective which describes the environment of the business process. The second one is the goal perspective which defines the expected outcome. Finally, we show how this model allows the designer to include semantics in his business process definition.

4.1 Context Perspective

In [17], the context is defined as a perspective that "provides an overview perspective of the process and describes major business process characteristics" for "people who do not know or do not need to know the process in detail". In this case, the context is some kind of summary perspective of the overall complex process. This is not the sense given to the word context in this paper. Rather, we agree with Aalst et al. in [3] where "the context describes the environment for which a process model has been designed" .

In this model, the context is captured by the notion of environment (defined as a set of observers) and the set of tasks. So the context is the set of tools available in a given environment.

For example, in a context of disaster such as after an earthquake, a medical doctor will not request an X-ray when he suspects a fracture, because there is probably no electricity, and even no radiologist around. The context determines the set of actions that can taken.

We can compare the process of treating a common tibia fracture in a normal context (BP_1) and in an emergency context (BP_2).

$BP_1 =< \theta_1, T_1, f_1, g_1 >$ and $BP_2 =< \theta_2, T_2, f_2, g_2 >$ where:
$\theta_1 = \{hasFracture\ Symptoms, isFracture\ Diagnosed,$
$isTibiaBroken, isXRayDone, isFracture\ Confirmed,$
$isCastApplied, isFracture\ Treated\}$
$T_1 = \{DiagnoseFracture\ , MakeXRay,$
$ConfirmFracture, ApplyCast\}$
$\theta_2 = \{hasFracture\ Symptoms, isFracture\ Diagnosed,$
$isTibiaBroken, isTibiaImmobilized, isFracture\ Treated\}$
$T_2 = \{DiagnoseFracture\ , ImmobilizeTibia\}$

In a normal context (BP_1), when a fracture is diagnosed, an X-Ray will be done to confirm the fracture before a cast is applied to treat the fracture. In an emergency context (BP_2), the task *ApplyCast* is replaced by $ImmobilizeTibia$ which can be done with a cast (if available) or any other means (like a piece of wood and a string to tie the leg). Also, the tasks $MakeXRay$ and $ConfirmFracture$ do not exist in the emergency context.

4.2 Semantics Perspective

When designing a business process, it is crucial to be able to also integrate semantic knowledge within the process [16]. The goal of integrating application knowledge is to enable the system to perform process checks at the semantic level.

This central preoccupation is tackled in this model by the notion of observation that enables one to define conditions.

Each observer of a business process environment is a semantic unit. Having it *true* or $false$ in a state yields a non ambiguous understanding that is captured by the notion of *observation* that we used to define the pre and post-condition of any business process task.

Using our task model, two semantic problems expressed in [16] are easily solved: mutual exclusion and dependency constraints.

Mutual exclusion constraints express that two activities are not compatible and should not be executed together. Mutual exclusion constraints are symmetric. For instance administering two incompatible drugs (Marcumar and Aspirineas as in [16]). We do not want a patient to take Aspirin after Marcumar.

Let us define a business process $BP =< \theta, T, f, g >$, where :

$\theta = \{tookAspirin, tookMarcumar, o, ...\}$
$T = \{AdministerMarcumar, AdministerAspirin, t_1, t_2, t_3, t_4\}$
$AdministerMarcumar$:
Pre-condition:
$P(ec(AdministerMarcumar\ )) = \{o\}$;
$M(ec(AdministerMarcumar)) = \{tookAspirin\}$;*(we want to make sure the patient did not take Aspirin before)*
Post-condition:
$P(AdministerMarcumar) = \{tookMarcumar\}$;
$M(AdministerMarcumar) = \{o\}$;
$AdministerAspirin$:
Pre-condition:
$P(ec(AdministerAspirin\ )) = \varnothing$;
$M(ec(AdministerAspirin)) = \{tookMarcumar, o\}$; *(we want to make sure the patient did not take Marcumar before)*
Post-condition:
$P(AdministerAspirin) = \{tookAspirin\}$;
$M(AdministerAspirin) = \varnothing$;
Tasks t_1, t_2, t_3, t_4 :
Pre-condition:
$P(ec(t_1)) = \varnothing; P(ec(t_2)) = \varnothing; P(ec(t_3)) = \varnothing; P(ec(t_4)) = \varnothing;$
$M(ec(t_1)) = \varnothing; M(ec(t_2)) = \varnothing; M(ec(t_3)) = \varnothing; M(ec(t_4)) = \varnothing;$
Post-condition:
$P(t_1) = \{o_1\}; P(t_2) = \{o_2\}; P(t_3) = \{o_3\}; P(t_4) = \{o_4\};$
$M(t_1) = \varnothing; M(t_2) = \varnothing; M(t_3) = \varnothing; M(t_4) = \varnothing;$
Follow function
$f(t_1) = \{AdministerAspirin, AdministerMarcumar\}$

With such a business process definition, we are assured that the patient will get either Marcumar or Aspirin, according to the value of the observer o that belongs to the pre-conditions of both tasks, but with opposite values. He will never get the two (fig. 2). This solution is the one found in [16].

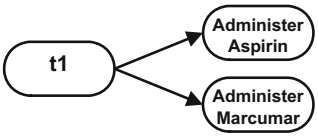


Fig. 2. Switch

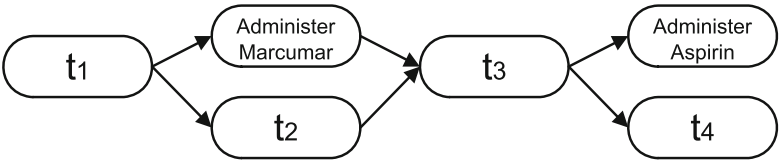


Fig. 3. Global dependency

Even with another follow function defined as:

$f(t_1) = \{t_2, AdministerMarcumar\}; f(AdministerMarcumar) = \{t_3\};$
$f(t_2\} = \{t_3\}; f(t_3) = \{AdministerAspirin, t_4\}$

If $AdministerMarcumar$ is the only task of the environment that modifies the observer $tookMarcumar$, we are assured that if $AdministerMarcumar$ has been executed, $AdministerAspirin$ will not be executed. This is because $AdministerMarcumar$ sets the observer $tookMarcumar$ to $true$, and no other task will change its value. It is then garanteed that $AdministerAspirin$ cannot be executed as it needs the observer $tookMarcumar$ to be $false$ as its pre-condition. This illustrates the global dependency property of the model. Two tasks do not need to be adjacent to enforce a dependency constraint.

This second solution (illustrated in fig. 3) were labelled conflictual in [16] since the conflict there was dependent of process structure, not on the semantics of tasks as it is the case here.

4.3 Goal Perspective

In this model, the notion of goal is explicit. In a business process $BP =< \theta, T, f, g >$, the goal g is an observation that has to be satisfied at the end of the process. Beyond the disambiguating role of the goal in the business process definition, it increases the reliability potential of the model.

Each task can be evaluated according to its input in the satisfaction of the goal (like counting the number of observers that the task sets for the goal). We then have two categories of useful tasks: those that directly impact the goal, and those that contribute to the satisfaction of the execution conditions of another task. For example, with the goal definition

$g : P(g) = \{isFracture\ Treated\}; M(g) = \varnothing,$

The execution of the task *ApplyCast* is enough to satisfy the goal; but to execute the task, its pre-condition requires that the observers $isFractureDiagnosed$ and $isFractureConfirmed$ be set to the value *true*. This is only possible if the tasks $DiagnoseFracture$ (to set the observer $isFractureDiagnosed$) and $ConfirmFracture$ (to set the observer $isFractureConfirmed$) have been executed. The task $ConfirmFracture$ itself needs the task $MakeXRay$ to be executed to set the observer $isXRayDone$ to true.

The notion of goal also enables the setting of a kind of process quality of service. The quality of service here do not refer to performance. It refers to the nature of the process. Using the previous section example, we can define goals with different quality of service.

$g : P(g) = \{isFracture\ Treated\}; M(g) = \varnothing$
g': $P(g') = \{isFracture\ Treated, isCastApplied\}; M(g') = \varnothing$

It appears that g' suggest a better quality of service than g, because unlike g, g' requires that the process designed respects particular care (applying a cast) that is considered essential for the quality of the final result.

5 Conclusion

In this paper, a business process modeling approach based on the concept of environment is presented, that allows one to precisely describe the execution context of business processes by using a set of observers. A task concept has been defined, where the action of every task is deterministic and the execution condition of a task is linked to the environment rather than a predifined and rigid order between tasks. Finally, given the environment and the tasks, a formal definition of a business process is given, that associates the environment and the tasks to a follow function in order to achieve a given goal. By expressing constraints over tasks using conditions over the environment rather than a strong task ordering scheme, tasks behave like independent components.

The model presented in this paper is used to address three perspectives: the context perspective, the semantic perspective and the goal perspective.

In ongoing researches, we are looking forward to enrich this model with business process generation and verification capacity given that semantics is easily expressed.

References

1. van der Aalst, W.M.P.: The application of Petri nets to workflow management. The Journal of Circuits, Systems and Computers 8(1), 21–66 (1998)
2. van der Aalst, W.M.P., van Hee, K.M.: Workflow management: models, methods, and systems, p. 267. MIT Press, Cambridge (2004)
3. van der Aalst, W.M.P., Rosemann, M., Dumas, M.: Deadline-based Escalation in Process-Aware Information Systems. Decision Support Systems 43(2), 492–511 (2007)

4. van der Aalst, W.M.P.: Three Good Reasons for Using a Petri-net-based Workflow Management System. In: Navathe, S., Wakayama, T. (eds.) IPIC 1996, Cambridge, Massachusetts, pp. 179–201 (1996)
5. Atsa Etoundi, R.: A Domain Engineering approach for multiperspectives Workflow modelling, Ph.D Thesis, University of Yaounde I - Cameroon (2004)
6. Atsa, R., Fouda, M.: An Abstract Model For Workflows and Business Processes. In: CARI 2002, pp. 239–247 (2002)
7. Atsa, R., Fouda, M.: Security Based Approach to Data Consistency in a Virtual Enterprise. In: ACS-IEEE International Conference on Computer Systems and Applications (2003), IEEE Catalog Number: 03EX722, ISBN:0-7803-7983-7
8. Atsa, R., Fouda, M.: Mobile-Based support for Business Processes: Feasibility and Correctness. In: ACS-IEEE International Conference on Computer Systems and Applications (2003), IEEE Catalog Number:03EX722, ISBN:0-7803-7983-7
9. Atsa, R., Fouda, M.: Human Resource Constraints driven Virtual Workflow Specification. In: Proceeding of the International Conference on Signal-Image technology & Internet-based Systems, pp. 176–182 (2005), 2-9525435-0 © IEEE SITIS 2005
10. Attie, P., Singh, M., Sheth, A., Rusinkiewicz, M.: Specifying and enforcing intertask dependencies. In: Proceedings of the 19th VLDB Conference (1993)
11. Cardoso, J., Sheth, A., Miller, J., Arnold, J., Kochut, K.: Quality of service for workflows and web service processes. Journal of Web Semantics 1(3), 281–308 (2004)
12. Davenport, T., Short, J.E.: The New Industrial Engineering: Information Technology and Business Process Redesign. Sloan Management Review, 11–27 (Summer 1990)
13. Dehnert, J., Freiheit, J., Zimmermann, A.: Modelling and evaluation of time aspects in business processes. Journal of the Operational Research Society 53, 1038–1047 (2002)
14. Kiepuszewski, P., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Fundamentals of Control Flow in Workflows. Acta Informatica 39(3), 143–209 (2003)
15. Lamport, L.: Specifying Concurrent Program Modules. ACM Transactions on Programming Languages and Systems 5(2), 190–222 (1983)
16. Ly, L.T., Rinderle, S., Dadam, P.: Semantic correctness in adaptive process management systems. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 193–208. Springer, Heidelberg (2006)
17. List, B., Korherr, B.: An Evaluation of Conceptual Business Process Modelling Languages. In: Proceedings of the 21st ACM Symposium on Applied Computing (SAC 2006). ACM Press, New York (2006)
18. Mendling, J., van der Aalst, W.M.P.: Towards EPC Semantics based on State and Context. In: Proceedings of the 5th EPC Workshop EPK 2006, CEUR 2006 Workshop Proceedings, pp. 25–48 (2006)
19. Motschnig-Pitrik, M.: Contexts as means to decompose Information Bases and represent relativized Information. In: Proc. CHI Workshop #11: The Who, What, Where, When, Why and How of Context-Awareness. Hague, Netherlands (2000)
20. Puhlmann, F.: Why do we actually need the Pi-Calculus for Business Process Management? In: Abramowicz, W., Mayr, H. (eds.) 9th International Conference on Business Information Systems (BIS 2006). Bonn, Gesellschaft fur Informatik. LNI, vol. P-85, pp. 77–89 (2006)
21. Trcka, N., van der Aalst, W.M.P., Sidorova, N.: Data-Flow anti-patterns: Discovering data-flow errors in Workflows. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 425–439. Springer, Heidelberg (2009)
22. Wenfei, F., Weinstein, S.: Specifying and Reasoning About Workflows with Path Constraint. In: Hui, L.C.K., Lee, D.-L. (eds.) ICSC 1999. LNCS, vol. 1749, pp. 226–235. Springer, Heidelberg (1999)

An Indexing Structure for Maintaining Configurable Process Models

Wassim Derguech, Gabriela Vulcu, and Sami Bhiri

DERI, Digital Enterprise Research Institute,
National University of Ireland, Galway
firstname.lastname@deri.org
http://www.deri.ie

Abstract. During the business process modeling phase, different environments and languages have been proposed. All of them are trying to narrow the communication gap between both business and IT users or making modeling task as optimal as possible. From these perspectives, we prioritize assisting business users to express in an efficient and easy way their requirements (i.e., defining their business process models). In this context, reusing existing process models is well supported and preferred rather than modeling from scratch. Configurable business process models aim at merging different process variants into a single configurable model. In this paper, we define an indexing structure to represent configurable process models. We give a set of structuring principles and we show how to maintain this structure when adding a new variant. We adopt a hierarchical representation of business goals variations. The contribution of our structure is that it allows for modularity handling.

Keywords: Business process, configuration-based modeling, goal hierarchy.

1 Introduction

Process Aware Information Systems (PAISs) [2] are used to manage and execute operational processes involving people, applications and data sources on the basis of business process models. The discipline that is concerned by this process-centric trend is known as Business Process Management (BPM) [15].

In Business Process Management the objective of the Business Process modeling phase is to capture the behavioural aspects (i.e., how to do it) of a certain business goal into a business process model [14]. There are several modeling approaches that can be split in two categories. The first one consists of designing business process models from scratch, which is an error prone and time consuming task [5]. The second category relies on reusing existing business process models.

The advent of Reuse-Oriented Development (ROD) in BPM brings a number frameworks used to support the design of business process models exploiting proven practices. One of these frameworks is the configurable process model.

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 157–168, 2010.

Configurable process models are constructed via the aggregation of several *variants* of a process model [12]. In fact, under different requirements, different business processes could achieve the same business goal. We call these business processes business process *variants*. Since they model in essence the same business goal, these variants share many commonalities. Therefore, managing these variants can be made easier by handling the common parts just once and not for each variant separately. In order to provide easier maintenance and management of variants, a key aspect of variability handling in process modeling is the explicit representation of *variation points*. A variation point is a special placeholder in the configurable process model in which variants are defined. During the business process modeling phase, the configurable process model is configured by setting up the variation points according to a user's specific requirements. These variation points capture different requirements that discriminate between the distinct parts of business process variants through *configuration parameters*.

To manage a configurable process model, we propose an indexing structure that captures variability at the business goal level. In this paper we define a hierarchical representation of configurable process models where a variation point is a business goal that has more than one business variant to achieve it. The rationale we opt for a hierarchical structure, which explicitly captures variation points, is to provide a user-friendly experience during the modeling phase while not overwhelming the modeler with cumbersome details from start.

Most of recent studies [1,3,4,6,10,11,12] consider managing business process variability under a single governance. However, in an open environment, managing variability with a top-down approach is not suitable. We propose an indexing structure that can be applied in such environments as it allows for defining new variation points by just adding a new variant of any business goal.

Our structure allows also for modularity handling. By modularity we mean the possibility to configure a sub process within the configurable process model. This is not possible with current approaches as the configuration phase consists of parsing the whole configuration process model.

The remainder of the paper is structured as follows. Section 2 describes a use case scenario to motivate the use of configurable process models in business process management. The indexing structure as well as its construction principle are presented in Section 3. Section 4 discusses some related work while Section 5 concludes the paper.

2 Motivating Example

In this section we give a motivating example for configuration-based modeling. The example describes a configurable process model in the job recruiting area as depicted in Fig. 1[1]. The process variability is modeled through a hierarchical structure of business goals (i.e., 'Find Possible Job Candidates' , 'Evaluate

[1] All figures in the paper are following the BPM notation: www.bpmn.org

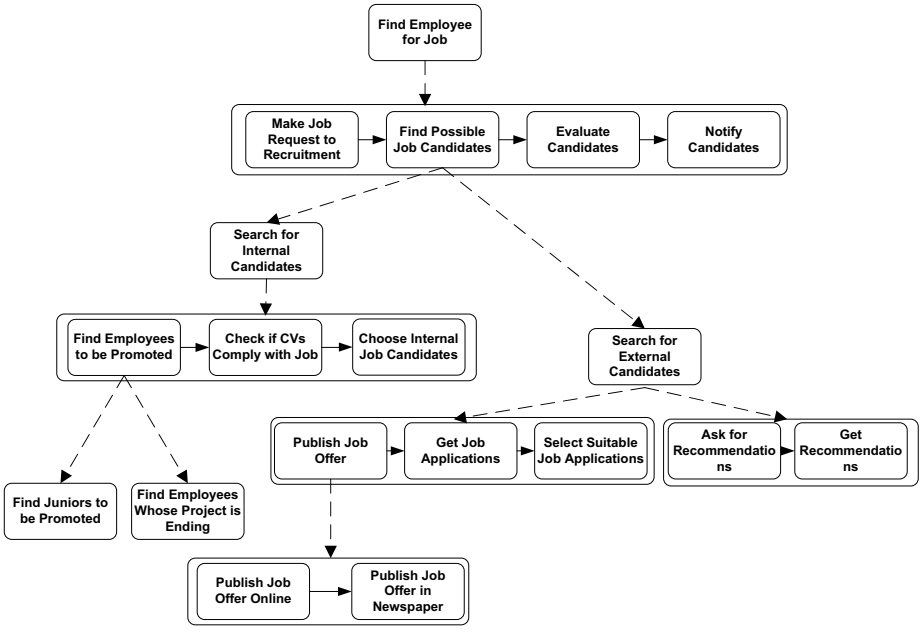


Fig. 1. The configuration process model from the job recruitment area

Candidates', 'Notify Candidates' etc.). In our model, we capture generalization-specialization relations between business goals. For example the 'Search for Internal Candidates' goal can be achieved at a lower layer of detail by a composed goal which integrates the 'Find Employees to be Promoted', 'Check if CVs Comply With Job' and 'Choose Internal Job Candidates' business goals in a sequence block pattern.

Variation points are modeled as business goals which can be achieved by many goals (simple or composite) one layer below in the hierarchy. This is represented in the Fig. 1 through the dotted line. For example, the 'Search for External Candidates' goal is a variation point because there are different ways of being achieved, either by publishing a job offer or by getting recommendations from acquaintances.

A typical scenario in this context is to find an employee in an enterprise for an available job, represented by the 'Find Employee for Job' business goal. We notice that there are many variants of such a process. At a certain moment, a business user would prefer a process that searches for internal job candidates(1), while in another context he would choose an external search(2). Normally, the business user would explore the configuration process model and, at the variation points, he would choose the most satisfiable alternative in a given context. Fig. 2 shows two possible variants.

Let us imagine that the configuration process model does not contain configuration parameters yet. In this situation configuration-based modeling is a

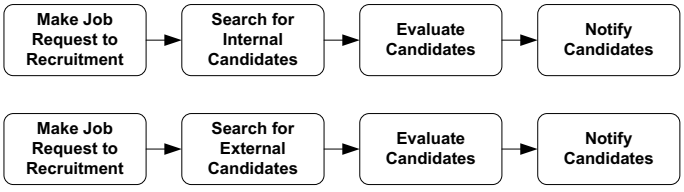


Fig. 2. Two different variants of 'Find Employee for Job' goal

cumbersome task for the business user because distinctive aspects are not captured in a user-friendly way in order to support the business user to make his selection.

That is why the need for well structured information represented by configuration parameters at variation points arises. Configuration parameters of the variation points represent a set of functional and nonfunctional aspects of all possible alternatives. For example, the configuration parameters of the 'Find Possible Job Candidates' business goal are the *time* with the options :[1-2 weeks] or [3-4 weeks] (non functional aspects) and the *candidateType* with the options: [internal] or [external](functional aspects).

In this paper we do not focus on presenting configuration parameters but rather on managing a data structure to capture the hierarchical representation of this goal hierarchy. The next section will introduce our reference model indexing structure.

3 Indexing Structure for Model Variations

In this section we define an indexing structure to represent configurable process models. We give a set of structuring principles and we show how to maintain this structure when adding a new variant. In our study we consider business goals as the main artifact of the configurable process model. We introduced the concept *abstract business goal* for manipulation purposes. A business process model is represented by at least one concrete business goal (i.e., it can be a sequence of business goals without any abstract one).

3.1 Indexing Structure Definition

We define a data structure to maintain configurable business process models. A configurable business process model is a tuple CPM=$\{\Sigma,\Gamma,\Delta\}$ where:

- Σ represents the set of business goals involved in the whole reference business process model,
- Γ represents the set of abstract business goals (will be introduced later in this paper) and
- Δ represents the possible variants of each business goal that can be a sequence of business goals. Entries of Δ are presented as:

$BusinessGoal_1 : BusinessGoal_2(-BusinessGoal_N)^*$ such that
$\{BusinessGoal_1, BusinessGoal_2, ...BusinessGoal_N\} \subseteq \Sigma \cup \Gamma$
and "$BusinessGoal_2 - BusinessGoal_3$" is the sequence between the two business goals.
This means that possible variants of BusinessGoal1 are presented as a sequence of other business goals. We call $BusinessGoal_1$ a *variation point* and $BusinessGoal_2(-BusinessGoal_N)^*$ a *variant*.

As an illustration, the motivating example of Fig. 1 would be presented as follows:

$CPM_1 = \{\Sigma_1, \Gamma_1, \Delta_1\}$ where:

- Σ_1 = { [Find Employee for Job], [Make Job Request to Recruitment], [Find Possible Job Candidates], [Evaluate Candidates], [Notify Candidates],[Search for Internal Candidates], [Find Employees to be Promoted], [Check if CVs Comply with Job], [Choose Internal Job Candidates], [Find Juniors to be Promoted], [Find Employees Whose Project is Ending], [Search for External Candidates], [Publish Job Offer], [Get Job Applications], [Select Suitable Job Applications], [Publish Job Offer Online], [Publish Job Offer in Newspaper], [Ask for Recommendations], [Get Recommendations] }
- Γ_1 ={ }
- Δ_1 = { [Find Employee for Job] : [Make Job Request to Recruitment] - [Find Possible Job Candidates] - [Evaluate Candidates] - [Notify Candidates],
  [Find Possible Job Candidates] : [Search for Internal Candidates],
  [Find Possible Job Candidates] : [Search for External Candidates],
  [Search for Internal Candidates] : [Find Employees to be Promoted] - [Check if CVs Comply with Job] - [Choose Internal Job Candidates],
  [Find Employees to be Promoted] : [Find Juniors to be Promoted],
  [Find Employees to be Promoted] : [Find Employees Whose Project is Ending],
  [Search for External Candidates] : [Publish Job Offer] - [Get Job Applications] - [Select Suitable Job Applications],
  [Publish Job Offer] : [Publish Job Offer Online],
  [Publish Job Offer] : [Publish Job Offer in Newspaper],
  [Search for External Candidates] : [Ask for Recommendations] - [Get Recommendations]}

As it was shown in Fig. 2, both variants for 'Find Employee for Job' can be generated from this structure. It allows also for managing modularity as we can configure even modules like 'Search for External Candidates' that has two variants. In the rest of the paper, for simplicity, we will use letter (e.g., A, B, C...) as labels for the business goals in order to avoid long strings.

3.2 Construction Principles

The indexing structure should respect a set of constraints/principles that assure it will remain valid and well-formed even after any update operation (i.e. adding a new variant).

We have identifies three principles: Minimality, Coverage and Consistency. They are presented here with a definition and an example.

Minimality

Definition: Each element of Δ has to be defined only once and should not be derived from other elements of Δ.

Example: $CPM_1 = \{\Sigma_1, \Gamma_1, \Delta_1\}$ where:

- $\Sigma_1 = \{A, B, C, D, E, F\}$
- $\Gamma_1 = \{\ \}$
- $\Delta_1 = \{A:B\text{-}C\text{-}D,\ A:E\text{-}F\text{-}C\text{-}D,\ B:E\text{-}F\}$

Being minimal imposes that Δ_1 represents the optimal representation of alternatives construction. With respect to this constraint, $\Delta 1$ should be $\Delta_1 = \{A:B\text{-}C\text{-}D,\ B:E\text{-}F\}$ (see Fig. 3).

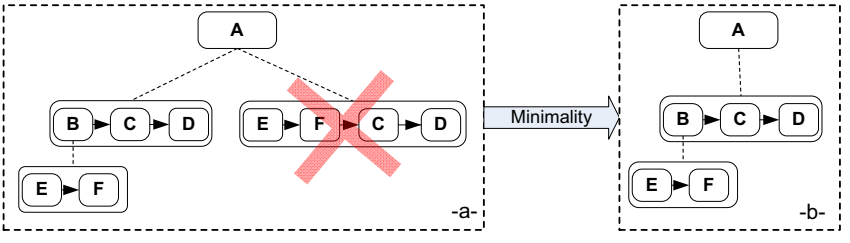


Fig. 3. The minimality principle

Coverage

Definition: : By necessity and nature, the indexing structure must cover all possible variants of the configurable business process model.

Example: In the example of Fig. 4 we can notice that possible variants generated from Δ (i.e., Fig. 4.a) into Fig. 4.b do not cover all possible variants from the Fig. 4.c.

Consistency

Definition: : Only defined variants should be deduced from the indexing structure and no extra ones are allowed to appear.

Example: : Considering that we have the list of defined variants in Fig. 5.a, the tree from the Fig. 5.b is wrong with respect to the set of defined variants. In fact we can deduce from it an extra variant which did not exist initially (e.g. B-G-H). The tree depicted in Fig. 5.c is a consistent structure.

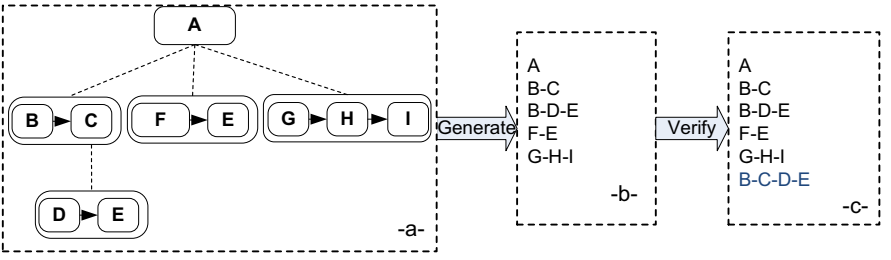


Fig. 4. The coverage principle

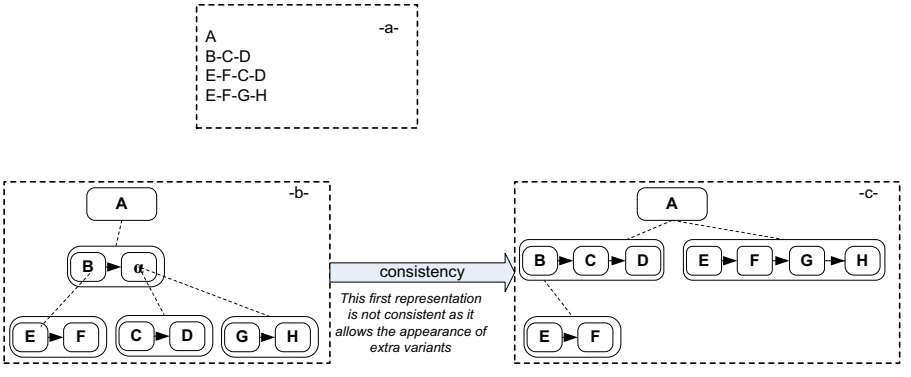


Fig. 5. The consistency principle

3.3 Maintaining the Indexing Structure When Adding a New Variant

When building the indexing structure, we start from a set of variants and we add them, one at a time, in the current structure while assuring the principles defined previously (see section 3.2) are not violated. When adding a new variant of a goal, the idea is to check, using matching detection, whether the variant (or parts of it) already exists in the indexing structure. There are three situations that may occur according to the matching degree between business goals of the new variant and those in the configurable business process model.

1. *Perfect match:* In this case the current variant to be inserted is entirely found in the current data structure. In such situation there is no action to be taken and the data structure remains as it.
2. *No matching:* In this case the current variant to be inserted is not found in the current data structure, not even partially. The variant is inserted as follows: all business goals composing the variant are added to Σ and the variant description is added to Δ.

Example: We want to insert the variant A:G-H-I in $CPM_1 = \{\Sigma_1,\Gamma_1,\Delta_1\}$ where:

- $\Sigma_1 = \{A, B, C, D, E, F\}$
- $\Gamma_1 = \{\ \}$
- $\Delta_1 = \{A:B\text{-}C, A:F\text{-}E, C:D\text{-}E\}$

The updated reference process model is then $CPM_1=\{\Sigma_1,\Gamma_1,\Delta_1\}$ where:

- $\Sigma_1 = \{A, B, C, D, E, F, G,H,I\}$
- $\Gamma_1 = \{\ \}$
- $\Delta_1 = \{A:B\text{-}C, A:F\text{-}E, C:D\text{-}E, A:G\text{-}H\text{-}I\}$

Using a tree representation, the variant is inserted as an alternative of the variation point as shown in Fig. 6[2].

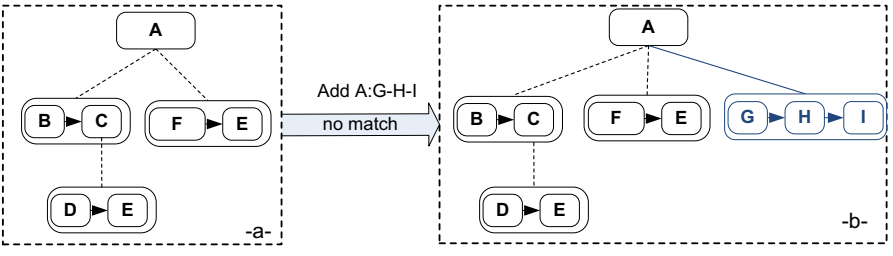


Fig. 6. The data structure (-a-) before and (-b-) after the insertion of a variant

3. *Partial match:* An intermediary situation is when a partial match occurs between the process variant to be inserted and the current data structure. In this case we distinguish two possible situations:

   - The first situation occurs when the new variant has common parts with another variant of the same business goal. A typical example is depicted in Fig. 7. This example shows adding the variant A:B-H-I in $CPM_1 = \{\Sigma_1,\Gamma_1,\Delta_1\}$ where:
     - $\Sigma_1 = \{A, B, C, D, E, F, G\}$
     - $\Gamma_1 = \{\ \}$
     - $\Delta_1 = \{A:B\text{-}C, A:F\text{-}G, C:D\text{-}E\}$

     The new variant A:B-H-I has B in common with A:B-C. To add this variant, an abstract business goal α is introduced to replace the different parts of these variants. The updated reference process model is then $CPM_1=\{\Sigma_1,\Gamma_1,\Delta_1\}$ where:
     - $\Sigma_1 = \{A, B, C, D, E, F, G, H, I\}$
     - $\Gamma_1 = \{\alpha\}$
     - $\Delta_1 = \{A:B\text{-}\alpha, \alpha:C, \alpha:H\text{-}I, A:F\text{-}G, C:D\text{-}E\}$

[2] All figures in the paper are following the BPM notation: www.bpmn.org

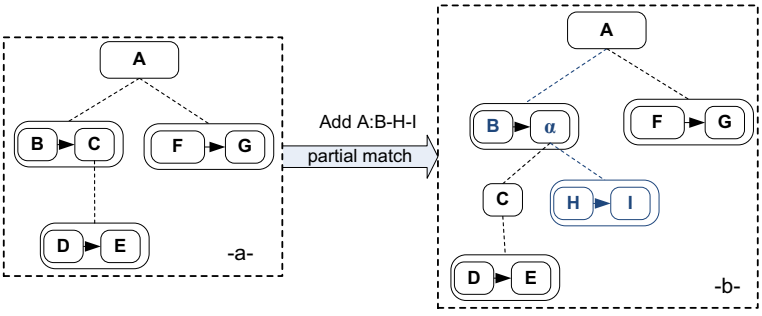


Fig. 7. The insertion of a variant including an abstract business goal

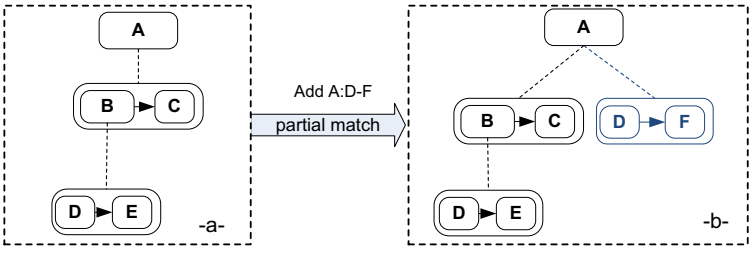


Fig. 8. The insertion of a variant with partial match without introducing an abstract business goal

– The second situation occurs when the new variant has common parts with another variant but not of the same business goal. A typical example is depicted in Fig. 8. This example shows adding the variant A:D-F in $CPM_1 = \{\Sigma_1,\Gamma_1,\Delta_1\}$ where:
  - Σ_1 = {A, B, C, D, E}
  - Γ_1 ={ }
  - Δ_1 = {A:B-C, B:D-E}

  This situation is similar to the second case (no match) and the updated reference process model is then CPM_1=$\{\Sigma_1,\Gamma_1,\Delta_1\}$ where:

  - Σ_1 = {A, B, C, D, E, F}
  - Γ_1 ={}
  - Δ_1 = {A:B-C, B:D-E,A:D-F}

4 Related Work

Several approaches have been proposed for defining and managing business process variants. In this section we state four current approaches dealing with process variability.

The first approach is the most intuitive solution to variability management. It consists of managing a repository of process variants. Each process model is stored as an individual entity in the repository. Users have to formulate a query according to their requirements and the system should provide the most suitable model. This approach has been explored by [7,8,9] where it reveals that it needs a rich formal model for describing business process. In our work, we do not use individual models because the main problems of this solution are resource allocation and inconsistency. Indeed, (i) storing each variant individually leads to duplicated data storage for common parts of the process models and (ii) in case of new regulations enforcement, all process variants have to be updated which is resource consuming and error prone task. In addition, variation points are not explicitly handled and in [7], configuration-based modeling relies on querying the process models repository based on structural aspects of the to-be process. Therefore the business user has to know what are the possible process structures he is allowed to ask for.

The second approach as it is presented in [4,3], overcomes the problems of resource allocation and inconsistency. This solution considers a "basic process model" that represents common parts of all process models and variability is handled as a global property containing a set of operations (e.g., add, delete, modify, move operation). In fact, each variant is then generated via applying these operations on the basic model. However, the business user's control becomes limited to a set of operations generating rules which fire when they comply with all the business requirements. These rules capture only non functional aspects (i.e., quality aspects like cost and performance) leaving out details about structural and functional aspects of the variants.

The third approach consists of generating a global flat process model containing all variations and each individual model is generated by eliminating some branches of the global model. [1,6] model process variability as explicit variation points within the control structure of a flat configurable model. However this solution poses visualization problems because, in a real world setting with a lot of process variants, the configurable process tends to get very large. Therefore the configuration model becomes difficult to comprehend and costly to maintain. But [11,12] reduced these problems by presenting a questionnaire-based configuration which is much more user-friendly than previous solutions.

In [11,12], the user specifies his business requirements by answering a set of domain-related questions. The authors distinguish between domain variability (i.e., it is based on domain facts which are features that can be enabled or disabled) and process variability (i.e., it is based on possible alternatives at a certain variation point). Both are related through a set of mappings such that the result of the domain-specific questions are reflected in the chosen alternative for a variation point. It is a very good option to make configuration user centric but IT experts are still highly needed to define both domain and model variability and their mapping which is manually performed and this makes the approach liable to subjectivity.

In addition, this approach is not flexible enough to manage modularity. Indeed, if the user wants to configure a particular business function that is embedded in

the global configurable model, he has to go through this model until reaching the intended business function to be configured. In our solution this problem cannot occur because we consider individual entities that can range from simple activities to complete process models.

The fourth approach studied in [10], is similar to ours as it exploits a hierarchical representation of the process into sub processes. The top level sub process encompasses the core activities and their associated variability, which is annotated by specific stereotypes, while the lower level sub processes express all details related to higher level activities and variabilities residing in them. However, the concept of hierarchical representation is supported more for hiding complexity than for managing variability.

5 Conclusion and Future Work

Configuration-based modeling is an important approach for business process management because it can decrease the modeling time and reduce the business user's work and risk to make errors. Configuration-based modeling is an iterative process of refinement actions in which the business user is assisted to lookup the most suitable process model depending on his requirements. Reviewing approaches that deal with configuration-based modeling, we have determined that they manage business process variability under a single governance as well as that they do not support modularity.

In this paper we proposed a structure for managing configurable process models. It is defined as a hierarchical indexing structure that captures process model's variability at the business goal level. We present a set of principles that the proposed indexing structure has to comply with and we show how it is maintained when adding a new variant to the configurable process model.

Our work is still in progress and continuous improvements are planned as a future work:

- We plan to formally define construction principles. New ones could be defined as well.
- A number of maintaining operations have not been yet explored or are still under definition, for example the deletion of a variant.
- We intend to investigate and extend this indexing structure in order to provide support for other block patterns (we have presented only sequence pattern in this paper).
- Our indexing structure is not exclusively designed for managing process variability. We eventually would experiment it in Mashups development environment to capture variability within Mashup applications.

Acknowledgment

This work is founded by the Lion II project supported by Science Foundation Ireland under grant number 08/CE/I1380.

References

1. Dreiling, A., Rosemann, M., van der Aalst, W.M.P., Sadiq, W., Khan, S.: Model-driven process configuration of enterprise systems. In: Ferstl, O.K., Sinz, E.J., Eckert, S., Isselhorst, T. (eds.) Wirtschaftsinformatik, pp. 687–706. Physica-Verlag, Heidelberg (2005)
2. Dumas, M., van der Aalst, W.M., ter Hofstede, A.H.: Process-Aware Information Systems: Bridging People and Software Through Process Technology. Wiley-Interscience, Hoboken (2005), ISBN: 0471663069
3. Hallerbach, A., Bauer, T., Reichert, M.: Context-based configuration of process variants. In: Proceedings of the 3rd International Workshop on Technologies for Context-Aware Business Process Management, TCoB (2008)
4. Hallerbach, A., Bauer, T., Reichert, M.: Managing process variants in the process life cycle. In: Cordeiro, J., Filipe, J. (eds.) ICEIS (3-2), pp. 154–161 (2008)
5. Indulska, M., Green, P., Recker, J., Rosemann, M.: Business process modeling: Perceived benefits. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., de Oliveira, J.P.M. (eds.) ER 2009. LNCS, vol. 5829, pp. 458–471. Springer, Heidelberg (2009)
6. Lapouchnian, A., Yu, Y., Mylopoulos, J.: Requirements-driven design and configuration management of business processes. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 246–261. Springer, Heidelberg (2007)
7. Lu, R., Sadiq, S.W.: On the discovery of preferred work practice through business process variants. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 165–180. Springer, Heidelberg (2007)
8. Markovic, I., Pereira, A.C.: Towards a formal framework for reuse in business process modeling. In: ter Hofstede, A.H.M., et al. (eds.) [13], pp. 484–495
9. Markovic, I., Pereira, A.C., Stojanovic, N.: A framework for querying in business process modelling. In: Bichler, M., Hess, T., Krcmar, H., Lechner, U., Matthes, F., Picot, A., Speitkamp, B., Wolf, P. (eds.) Multikonferenz Wirtschaftsinformatik. GITO-Verlag, Berlin (2008)
10. Razavian, M., Khosravi, R.: Modeling variability in business process models using uml. In: ITNG, pp. 82–87. IEEE Computer Society, Los Alamitos (2008)
11. Rosa, M.L., Lux, J., Seidel, S., Dumas, M., ter Hofstede, A.H.M.: Questionnaire-driven configuration of reference process models. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 424–438. Springer, Heidelberg (2007)
12. Rosemann, M., van der Aalst, W.M.P.: A configurable reference modelling language. Inf. Syst., 32(1):1–23 (2007)
13. ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.): BPM Workshops 2007. LNCS, vol. 4928, pp. 66–77. Springer, Heidelberg (2008)
14. van der Aalst, W.M.P.: Business process management. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, pp. 289–293. Springer, US (2009)
15. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer, Heidelberg (2007)

A Meta-language for EA Information Modeling – State-of-the-Art and Requirements Elicitation

Sabine Buckl, Florian Matthes, and Christian M. Schweda

Chair for Software Engineering of Business Information Systems (sebis),
Technische Universität München,
Boltzmannstr. 3, 85748 Garching, Germany
{buckls,matthes,schweda}@in.tum.de
http://wwwmatthes.in.tum.de

Abstract. Enterprise architecture (EA) management has not only recently gained importance as means to support enterprises in adapting to a changing market environment and in seizing new business opportunities. This twofold role of EA management in transforming enterprises is connected to describing the current state as well as future states of the EA. Although different information models for the description of these states have yet been proposed in literature, no 'standard' information model exists, and the plurality advocates for the idea that such models are enterprise-specific design artifacts.

In this paper, we explore the fundamentals of EA information modeling, namely the meta-languages underlying today's models, and analyze their diversity. Based on the analysis, we elicit requirements for a "unifying" meta-language. By showing that multi-purpose modeling facilities, as the OMG's UML, fail to fully satisfy these requirements, we establish a future field of research – a meta-language for EA information modeling.

Keywords: Enterprise architecture, Modeling language.

1 Motivation

Adapting to changes of the environment is a critical success factor for today's enterprises. An instrument, which is commonly regarded to be supportive in this context, is the management of the enterprise architecture (EA). Architecture, in this context, is understood as the "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" [1]. Therefore, EA provides a holistic perspective on the enterprise and its constituents as well as on the environment, and considers business as well as IT-related aspects. The goal of EA management is to provide support for organizational change and enhance the alignment of business and IT [2,3]. Major tasks of EA management are the description and analysis of the current EA state as well as providing support for the planned evolution of the architecture via comparing future scenarios [4,5], and selecting the project portfolio guiding the transformation [6].

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 169–181, 2010.

Although the topic of EA management has been under research by a large community of practitioners [7], academic researchers [8,9,10,11], public authorities [12], and tool vendors [13], since the late 70ies, no commonly accepted standard has yet emerged. The absence of such a standard becomes apparent, if architectural descriptions are considered. Various modeling techniques and complementing information models[1], providing the terms and concepts necessary for describing an EA, have been proposed, which differ widely in respect to the concepts that they employ as well as the coverage of the EA that they aim at [8]. The plurality of available models advocates for the idea of such models being organization-specific design artifacts. This fact is backed by techniques, which describe how an enterprise-specific information model can be developed [8,11].

Albeit the non-existence of a standard information model, which can be ascribed to the enterprise-specificity of such artifact, the meta-language for describing information models seems to be a candidate for more detailed discussions. In particular, one can expect the requirements for such a meta-language not to be organization-specific. We will see some indications supporting the former hypothesis in Section 2, when we revisit the state-of-the-art in EA information modeling. Nevertheless, the models presented in the EA management approaches listed above, differ widely in respect to the underlying meta-languages. Two possible reasons might cause this plurality:

- The information models differ strongly in respect to the grounding abstractions of the modeling domain or
- the information models were developed independently on arbitrarily chosen conceptualizations for describing the modeling domain.

The truth may most likely lay somewhere between these extremes and poses an interesting subject for in-depth research. This is especially true, as the lack of a single dedicated meta-language for EA information modeling hampers the advance in this field of research. On the one hand, different information models are hardly comparable, if they utilize different meta-languages. On the other hand, models grounded on different meta-languages cannot easily be combined into a comprehensive model, if a using enterprise would like to leverage the advantages of the individual models. Finally, some of the aforementioned EA management approaches bring along information models that are grounded in meta-languages that have originally been developed for other purposes. A prominent example for such a meta-language is the UML [14]. This utilization of general purpose meta-languages tends to result in misusing concepts and in developing isolated meta-language extensions (cf. [15]). Furthermore, most of these meta-languages provide specialized concepts for the originally intended usage scenario, which are not needed for a meta-language for EA information modeling. This may lead to the creation of unintended models, i.e. models that use the language concepts in a way not intended for the modeling domain.

On this background, we regard the topic of a meta-language for EA information modeling to be an open issue. This article approaches the experienced gap

[1] In line with the terminology of Buckl et al. (cf. [8]), we use the term "information model" when referring to the meta-model used for EA modeling.

in a twofold way. In Section 2, we revisit the state-of-the-art in EA information modeling with special emphasis on the meta-languages used to build the corresponding information models. Taking the general perspective of a multi-purpose ontology [16], we propose a set of requirements for a meta-language for EA information modeling in Section 3 and link the requirements back to the underlying statements from EA-related literature. Section 4 sketches how currently used modeling facilities fail to completely fulfill the requirements, and concludes with an outlook on future research ideas in the context.

2 Meta-languages Used for EA Information Modeling

In this section, we analyze the meta-languages underlying prominent EA information modeling approaches. The selection of approaches is based on a literature survey of Aier et al. [17], and the analysis of Schelp and Winter [18].

The Archimate approach to enterprise modeling is among others presented by Jonkers et al. [19] and was further refined in [20]. In these publications, the relevant concepts for describing an EA are introduced using a somewhat intuitive notation. Put in other words, the Archimate information models are presented in a proprietary diagrammatic notation without explicit reference to an underlying meta-language. Only a side-note grants a glimpse on the meta-language, more precisely states, that two distinct types `Thing` and (binary) `Relation` are contained therein. Whereas, other concepts, as e.g. `Properties`, are not directly introduced, there is evidence that an identifying `name` property is associated with every `Thing`. Complementing the information model, Jonkers et al. provide a dictionary of terms, i.e. textually describe the meanings of the concepts introduced in the model.

From an information model point of view, a central publication of the St. Gallen approach to EA management is the "core business metamodel" as introduced by Österle et al. [21]. The publication names the UML [14] as the underlying meta-language of the information model, more precisely, only a "pragmatic" subset of thereof [22]. Mainly, the concepts `Class` and (binary) `Association` are used, while in occasional cases the associations are refined to `Aggregations` or `Association Classes`. Aggregations are thereby often used to describe hierarchies in the information model. Finally, `Generalizations` and `Specializations`, respectively, are used to build the information model of the St. Gallen approach. No further concepts from the UML are employed, such that the meta-model most evidently lacks the capabilities to specify `Properties` owned by the classes as well as `Multiplicities` for constraining the valid instantiations. This fact is nevertheless partially mediated as in instantiations of the core business metamodel or parts thereof the corresponding objects are evidently `named`.

The EAM approach developed at the Royal Institute of Technology in Stockholm, more precisely details of the underlying information modeling technique, are described in the group's book on EA analysis [5]. It has to be emphasized that the approach employs two different modeling techniques – one underlying the relevant "EA viewpoints", as they call the information models, and a different one backing their analysis models. The former technique, more precisely the

underlying meta-model is not explicitly alluded to, but its concepts strongly resemble UML concepts as `Class` and (binary) `Association`, where associations are further constrained via `Multiplicities`. The models also employ the concepts of `Generalization` and `Specialization`, respectively, while prominently the concept of `Property` is omitted. Again, evidence exists that at least a `name` property is supplied with every information model class. Properties play also an important role in the meta-model backing the analysis models, denoted as "influence diagrams" in the approach. Influence diagrams are used to operationalize EA-relevant goals to "abstract qualities" that are further detailed towards observable qualities or properties related to architectural elements. In this sense, the meta-model of the influence diagrams introduces the concept of `Goal` and `Quality/Property` as well as two types of relationships: `Definitional Relationships` and `Causal Relationships`, of which the later describe that a change in one property is most likely to cause the change of a related property.

The Open Group Architecture Framework (TOGAF) in its most recent version 9 [7] provides the "architecture content framework", which is grounded in a "content metamodel" introducing the relevant concepts. The meta-language underlying the TOGAF's information model is not explicitly alluded to, but the employed terminology points towards the utilization of an object-oriented meta-language as the UML [14]. More precisely, a subset of the UML concepts is used, namely `Class`, `Property`, and (binary) `Association`. Further, the mechanisms of `Generalization` and `Specialization` are used. Other concepts from the UML are not employed, such that evidently `Multiplicities` for the associations and `typing` for the attributes are missing. Special to TOGAF's "content metamodel" is some sort of packaging mechanism that partitions the information model into six distinct units "Core", "Process", "Governance", "Motivation", "Data Modeling", and "Infrastructure Consolidation". While no precise semantics of the packaging mechanism are given in TOGAF [7], the provided examples exert strong similarities with the UML `Package Merge`, i.e. a mechanism that allows to provide additional specification for one class in a package different from the package, where the class was initially defined. Complementing the information model, TOGAF provides a comprehensive dictionary textually defining both the classes and the properties used in the "content metamodel".

As part of the Systemic Enterprise Architecture Methodology (SEAM), Lê and Wegmann provide in [23] an "object-oriented modeling language for EA". According to the text, the information model's underlying meta-language is the UML [14], of whose concepts conversely only a subset is used. Putting it more precisely, the information model builds on the concepts of `Class` and (binary) `Association`, respectively, of which the latter can further be refined to `Compositions` or via `Association Classes`. Compositions are used as means to impose hierarchies in the architecture model. Also the mechanisms of `Generalization` and `Specialization` are used. Finally, the associations are constrained by `Multiplicities` and by additional formal `Constraints` supplemented in a set-theoretic language. Abstaining from utilizing other concepts of the UML, the meta-language lacks the concept of `Property`. Complementing the

information model, Lê and Wegmann (cf. [23]) textually describe the semantics of the information model classes.

The Multi-Perspective Enterprise Modeling (MEMO) approach introduced by Frank in [24] and refined over the years in multiple publications (see among others [25,26]) puts critical emphasis on the topic of modeling. In this light, it is not surprising that the approach not only brings along a comprehensive set of information models underlying the different special-purpose modeling languages, which constitute the approach. MEMO also explicitly alludes to the meta-language (MML), on which the different MEMO languages are built. This meta-language is described by Frank in [9] and presents itself as an object-oriented language comprised of the conceptions of `MetaEntity`, `MetaAttribute`, and `MetaAssociationLink`[2]. Both properties and associations supply the concept of `Multiplicity`, whereas properties are further *strongly-typed* in a domain-agnostic type-system consisting of basic `Datatypes`. Classes in the MML can be related via the mechanisms of `Generalization` and `Specialization`, and can further be flagged as `abstract`. Beside the rather basic concept of the `Constraint` expressed in an OCL-like syntax [27], the MML supports the sophisticated notion of the `intrinsic` concept. This concept plays a crucial role, when multiple levels of *metaization* are considered. For example an intrinsic property specified on meta-level $n + 2$ is linguistically instantiated into a "normal" property on meta-level $n + 1$ and can finally be assigned to values on meta-level n. Intrinsic concepts resemble a *potency* of limited depth as introduced by Atkinson and Kühne in [28]. To round up the analysis of the MEMO approach, we should have a closer look on the ScoreML, a special-purpose modeling language outlined by Frank et al. in [26]. The conceptual model of ScoreML introduces the notion of "goal" and decomposes the model towards operationalized "performance indicators". These indicators are conversely associated to classes and concepts from an arbitrary special purpose language for the corresponding relevant part of the overall EA. In the ScoreML, further concepts for defining and relating "performance indicators" are supplied.

The pattern-based approach to EA management as presented in the EAM pattern catalog of Technische Universität München [29] presents a set of information model fragments, called *I-pattern*, for modeling EAs. These fragments use the UML as meta-language, more precisely concepts for describing static aspects as also reflected in the Meta-Object Facility (MOF) [30]. The approach uses abstract and non-abstract `Classes`, *typed* `Properties`, and (binary) `Associations`, which can further be refined to `Compositions` and via `Association Classes`. For typing the properties, domain-specific `Datatypes`, as e.g. `Money` are used where necessary, but lack a comprehensive definition. The mechanisms of `Generalization` and `Specialization`, `Multiplicities` for both properties and associations, and `Constraints` in OCL syntax [27] are used throughout the information model fragments. In addition, selected information model fragments represent relevant architecture performance indicators via `derived` properties that are also supplied

[2] Translated to the terminology of the UML, the three concepts denote `Classes`, `Properties`, and `AssociationEnds`, respectively.

with *derivation rules* expressed in OCL. Annotating the information model fragments, the EAM pattern catalog [29] further supplies textual descriptions for the classes' meanings. In a joint publication of KTH Stockholm, TU Berlin, and Technische Universität München [31] Buckl et al. discuss along a practical example how the MOF can be extended to support goal-specific dependency modeling in the EA. In this context the "definitional dependencies" are introduced to the meta-language provided by MOF (cf. also Buckl et al. [15]).

Complementing the forestanding analysis of the state-of-the-art in information models for describing EAs as found in literature, we further analyze the models and underlying meta-languages as used in today's prominent EA management tools. Abstaining from enumerating the very details of the meta-language tool-by-tool, we summarize common characteristics here and come back to "exotic" characteristics, when eliciting the meta-language requirements in Section 3. The most prominent tools as analyzed by Matthes et al. in [13] build on an object-oriented meta-language comprised of the concepts `Class`, `Property`, and (binary) `Association`, while the latter concept is sometimes substituted by *mutual* properties. Further, the tools only support *strongly-typed* properties, often providing a rich set of domain-appropriate `Datatypes` as *money* or *date*. Rounding up this short exposition of common characteristics of the tool's meta-languages, we can say that the mechanisms of `Generalization` and `Specialization` are widely supported.

3 Requirements

Based on our findings from Section 2, we present requirements for a meta-language for EA information modeling. The subsequent list is thereby not meant to be exhaustive, but delineates requirements that can be grounded well in existing literature on EA management. In this respect, we aim at presenting the most relevant of these requirements as basis for a subsequent analysis of the suitability of multi-purpose modeling facilities. To illustrate our requirements, we give, where possible, illustrative object-oriented models (using the UML[3] [30]), reflecting a typical situation, in which the corresponding requirement applies.

(R1) ***Modeling primitives.*** The different EA information models are built on a small set of modeling primitives that conversely must be supported by the corresponding meta-language. Most prominently, these primitives are `Classes`, typed `Properties`, and binary `Associations`. On both properties and associations prominently multiplicity constraints apply, i.e. a *lower bound* and an *upper bound* can be specified. In particular, one must have the chance to express that a property is *mandatory*. Reflecting the forestanding primitives against the background of Guizzardi's ontology [16], we could rise the question, whether a `name`-property should be specifically accounted for. In the sense of Guizzardi, any "thing", i.e. instance of class in our terms, has an *identifying* property. In the context of EA information modeling, we could sensibly assume that the name of a thing would be

[3] The utilization of the UML should not be misinterpreted as statement to use UML as meta-language for EA information modeling. We nevertheless found it a both commonly used and convenient language for describing object-oriented models of any kind.

such property. Delving deeper into ontological subtleties, we could ask on the exact understanding of "class". More precisely, we could generalize the class concept towards the concept of the "universal", as discussed by Guizzardi in [16]. For reasons of brevity, we abstain from detailing such considerations here, which would nevertheless be beneficial to refine the modeling primitives. The same applies for the subtleties of `Generalization` mechanisms, whereas basic inheritance must be supported by the meta-language.

As a lightweight counterpart for classes, the meta-language must also supply a `Datatype` concept complemented with a set of basic data types reflecting typical EA related concepts as *money*, *time*, or *probabilities*. In extension to this, an `Enumeration` concept is needed in the meta-language to specify domain limitations for certain properties, see e.g. [8,32,33]. In respect to associations, the meta-language must support a concept to *reify* associations, i.e. understand instantiated associations as "things" that themselves can have properties and associations again. The need to reify associations is clearly supported by the models of Österle [21] and Kurpjuweit, as well as the ones found in the EA management pattern catalog of Technische Universität München [29], which all use the UML concept of the `Association Class`. In line with the argumentation of Guizzardi in [16], a `Relator`-concept should be used to reify an assocation providing a clear distinction between relationship and thing-nature of an element.

(R2) ***Hierarchy modeling.*** Hierarchies are prominently used throughout modeling EAs. Thereby, the models reflect hierarchic, i.e. tree-like, structures in the real-world enterprise, e.g. organizational structures, business process hierarchies, or the architecture of component-based business applications. In structures, like the aforementioned ones, the outgoing relationships of *supernodes*, i.e. elements on higher hierarchy-level, are derived from their corresponding *subnodes*. A typical model fragment, illustrating such hierarchy modeling is shown in Figure 1, although pure UML is not sufficient to clearly constrain the model to a hierarchy. To achieve this, further constraints, e.g. using the OCL [27] would be necessary to demand that the parent-child-relationships and its transitive closure, respectively, are acyclic. Examples of hierarchy modeling can be found in different EA management approaches, e.g. the approach presented by Fischer and Winter [34]. According to Matthes et al. [13], many of the currently available EA management tools support hierarchy modeling.

Resorting to the ontological foundations presented by Guizzardi in [16], modeling hierarchies can be regarded a special case of *whole-part-relationships*. The corresponding ontological discipline of *mereology* presents a broad field of possible properties that whole-part-relationships may have. Especially the question, if such a relationship is transitive, would deserve special attention. We abstain from in-depth considerations on this topic here, which may in accordance to Kurpjuweit [35] be also relevant in the context of EA information modeling.

(R3) ***Constraint specification.*** The meta-language must support language concepts for specifying quantified mathematical and logical expressions over the information model, acting as constraints in model instantiation. While as far as

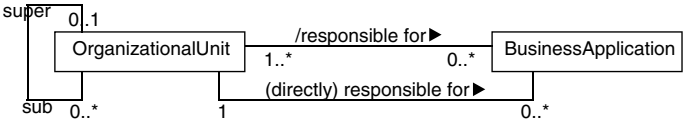


Fig. 1. Object-oriented model of a hierarchy

possible, the concepts of the meta-language itself should supply mechanisms that restrict their instantiation in order to prevent *unintended models*, there might exist multiple domain-specific constraints that cannot be incorporated in terms of e.g. multiplicities. A simplistic example for such domain-inherent constraint is described in the EA management pattern catalog [29], where a constraint is used to demand that a project starts before it ends (`startDate < endDate`).

(R4) ***Dependency explication.*** In the context of EA management, many authors, e.g. Niemann [36], express that architectural descriptions are mostly about the relationships between the architectural elements. Dependency explication extends the simple understanding of the relationships towards a more dynamic notion of relation, i.e. dependency, where the EA model can express that an architectural property of one concept is dependent on architectural properties of related concepts. Such dependency modeling can take the simple form of *rules for derivation* as presented by Lankes and Schweda [33] or Frank et al. [26]. But also more complex *cause-effect relationships* between architectural properties may exist, reflecting the behavioral dynamics of the EA. These dependencies are accounted for by different relevant approaches in the field of EA management, e.g. the ones of Buckl et al. [31], Johnson and Ekstedt [5] or of Yu et al. [37]. Dependency modeling can further be understood as generalization of transitive relationship modeling as presented by van Buuren et al. in [38].

(R5) ***Multi-level modeling.*** The demand for multi-level modeling applies to many fields in which – speaking in terms of Guizzardi's ontology, cf. [16] – *things* and their corresponding *sortals* should be modeled simultaneously. A related discussion is undertaken by Engelbert and Heymans in [39]. To exemplify the demand for multi-level modeling in the context of EA management, we present a typical *type-item* pattern found in an information model for EA management in the EAM Pattern Catalog [40], see Figure 2. The information model Technology and Connector Usage is used to model architectural standardization on the one hand on the level of architectural guidelines (e.g. three-tier architecture) and on the other hand on the level of actual technologies and technology stacks. This leads to the typical type-item dichotomy that can be found multiple times in other EA models, e.g. described by Matthes et al. [13] or Frank [24]. The problems of modeling the type-item dichotomy by using object-oriented means, become apparent not only along the duplication of concepts, but also with the demand to add further constraints to ensure modeling consistency.

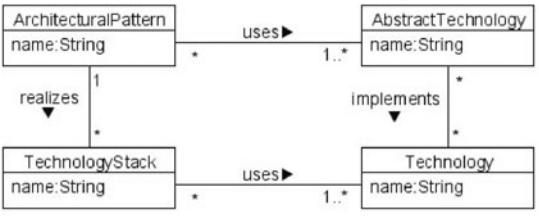


Fig. 2. EA information model exemplifying the type-item pattern

(R6) ***Packaging and package relationship mechanisms.*** Relating back to the stakeholder-centric perspective on architectural modeling as advocated by the ISO standard 42010 [1], it seems sensible to decompose the overall architecture into different *areas of interest*, which reflect the stakeholders' architectural concerns. These concerns do nevertheless not exist in isolation, but relate to each other in manifold ways. This can be illustrated along a simple example describing the relationships between the business applications in an enterprise. While a software architect might only be interested to know about the relationships between the application, an infrastructure architect might have a more detailed concern in this respect, needing additional information on the used information exchange protocols, etc. In this example, one might sensibly say that the software architect's concern is totally included in the infrastructure architect's one. The different concerns of the stakeholders thereby correspond to their *base-level* of architectural knowledge, such that in quite some cases inclusion relationships as the one illustrated above, may be derivable. A prominent example for relationships of that type can be found in the stakeholder-oriented approach to EA management presented by Aier in [11]. Similarly, the pattern-based approach to EA management presented by Ernst in [41] establishes relationships between the different information model patterns, reflecting relationships in their represented concerns. In this vein, the types of relationships between patterns as discussed by Noble in [42] may serve as basis for defining the relevant classes of relationships between EA information model fragments. This modeling of relationships between model fragments is further advocated in the discussions of Kurpjuweit and Aier in [22], where they propose to utilize a *composition operator* to consistently aggregate architectural model concepts. The authors argue that the thereby abstracted models are useful for creating EA descriptions, as the modeler is not forced to specify intermediary concepts, if he does not have knowledge about them, i.e. allow for switching the base-level perspective.

(R7) ***Intentional semantics.*** EA models, i.e. instantiations based on EA information models, are used as means to support communication among different interest groups in an organization. The employees in these interest groups most likely have differing educational backgrounds and may use a different terminology in respect to the enterprise. The EA information models target to comprehensively describe the "universe of discourse", i.e. the relevant parts of the enterprise, and hence may fall for ambiguities concerning the understanding of the used terms. To prevent communication issues, the meta-language must provide techniques and

mechanisms suitable for describing the meaning of the modeled elements. In this sense, two generally different approaches can be distinguished. The meta-language may allow to supply a description for each modeled element. In line with the understanding of Kamlah and Lorenzen [43], such method would shape a *linguistic community* embracing all relevant EA stakeholders. An alternative approach would comprise mechanisms to supply stakeholder-specific descriptions and names for the modeled elements. This approach accounts for the linguistic plurality in an enterprise, and provided a distinct and consistent terminology for every relevant linguistic community in the enterprise.

4 Summary and Outlook

Summarizing, we can state that mostly two meta-languages are used to build EA information models, namely UML (or more precisely subsets of the UML infrastructure) and the special purpose MML of Frank [9]. Table 1 shows how the two meta-languages fulfill the requirements specified in Section 3. The fulfillment of each requirement ranges from nearly complete fulfillment (●) via partial fulfillment (◑) to complete lack of support (○).

Forestanding Table 1 indicates that as-of-today none of the used meta-languages fully satisfies the elicited requirements. While one might argue that this may ascribe to the fact that yet no such language was needed, we take a different position. The absence of clear references to the underlying meta-language in many of the EA information modeling approaches outlined in Section 2 seems to us an indication towards the missing engagement in this field.

Our paper does not present a comprehensive meta-language for EA information modeling, nor does it claim to present an embracing set of requirements for such a language. The requirements presented in Section 3 in contrast formulate a "base

Table 1. Comparison of possible meta-languages for EA information modeling

	R1[1]	R2[2]	R3	R4[3]	R5	R6	R7[8]
MML + OCL + ScoreML	◑	◑	●	◑	◑[4]	○[6]	○
UML (infrastructure) + OCL	◑	◑	●	◑	○[5]	◑[7]	○

[1] Both UML and MML do not support domain-specific datatypes as money or date.
[2] Hierarchies can be modeled using additional constraints in OCL.
[3] OCL allows to specify and operationalize dependencies but does not support pure specification without derivation rule.
[4] The MML provides the notion of the "intrinsic" feature, that allows two-level instantiation.
[5] The UML follows a strict class-object-dichotomy, i.e. a single-level instantiation.
[6] The MML only supplies a simple packaging mechanism without package composition.
[7] The UML package merge allows model element re-use on class level.
[8] Both UML and MML do not supply a mechanism for specifying the meaning of a concept.

line" for any meta-language for EA information modeling, retaining potential for future extensions. Especially, the aspect of *uncertainty* (cf. Johnson et al. [44] or Aier et al. [45]) as well as the aspect of *temporality*, as discussed by Buckl et al. in [6], may be of relevance for a meta-language. In the context of temporality also the question of *non-rigid* typing, see e.g. Guizzardi [16], may play an important role and lead to additional requirements.

The findings of the paper may – notwithstanding the aforementioned limitation – provide substantial input for the development of the topic, i.e. for finding or designing a domain appropriate meta-language for EA information modeling. We see such language as very beneficial for the overall advancement of the field, as clear and concise modeling of relevant concepts may allow to compare and relate the different models proposed in the EA management approaches. Finally, a specialized meta-language could lay the basis for a toolset for EA information modeling that should also be valuable for implementing and supporting EA management functions in practical environments.

References

1. International Organization for Standardization: Iso/iec 42010:2007 systems and software engineering – recommended practice for architectural description of software-intensive systems (2007)
2. Henderson, J.C., Venkatraman, N.: Strategic alignment: leveraging information technology for transforming organizations. IBM Systems Journal 32(1), 472–484 (1993)
3. Luftman, J.N.: Competing in the Information Age – Align in the Sand, 2nd edn. Oxford University Press, New York (2003)
4. Holschke, O., Närman, P., Flores, W.R., Eriksson, E., Schönherr, M.: Using enterprise architecture models and bayesian belief networks for failure impact analysis. In: Aier, S., Johnson, P., Schelp, J. (eds.) Pre-Proceedings of the 3rd Workshop on Trends in Enterprise Architecture Research, Sydney, Australia, pp. 33–46 (2008)
5. Johnson, P., Ekstedt, M.: Enterprise Architecture – Models and Analyses for Information Systems Decision Making. Studentlitteratur, Pozkal, Poland (2007)
6. Buckl, S., Ernst, A.M., Matthes, F., Schweda, C.: An information model for managed application landscape evolution. Journal of Enterprise Architecture (JEA) 5(1), 12–26 (2009)
7. The Open Group: TOGAF "Enterprise Edition" Version 9 (2009), http://www.togaf.org (cited 2010-02-25)
8. Buckl, S., Ernst, A.M., Lankes, J., Schneider, K., Schweda, C.M.: A pattern based approach for constructing enterprise architecture management information models. In: Wirtschaftsinformatik 2007, Karlsruhe, Germany, pp. 145–162. Universitätsverlag Karlsruhe (2007)
9. Frank, U.: The memo meta modelling language (mml) and language architecture (icb-research report). Technical report, Institut für Informatik und Wirtschaftsinformatik, Duisburg-Essen, Germany (2009)
10. Lankhorst, M.: Introduction to enterprise architecture. In: Enterprise Architecture at Work. Springer, Heidelberg (2005)
11. Aier, S., Kurpjuweit, S., Riege, C., Saat, J.: Stakeholderorientierte documentation und analyse der unternehmensarchitektur. In: Hegering, H.G., Lehmann, A., Ohlbach, H.J., Scheideler, C. (eds.) GI Jahrestagung (2), Bonn, Germany, Gesellschaft für Informatik. LNI, vol. 134, pp. 559–565 (2008)

12. NATO: Nato architecture framework version 3 (2007), http://www.nhqc3s.nato.int/ARCHITECTURE/_docs/NAF_v3/ANNEX1.pdf (cited 2010-02-25)
13. Matthes, F., Buckl, S., Leitel, J., Schweda, C.M.: Enterprise Architecture Management Tool Survey 2008. Chair for Informatics 19 (sebis), Technische Universität München, Munich, Germany (2008)
14. Object Management Group (OMG): Uml 2.2 superstructure specification (formal/2009-02-02) (2009), http://www.uml.org (cited 2010-02-25)
15. Buckl, S., Ernst, A.M., Schweda, C.M.: An extension to the essential meta-object facility (emof) for specifying and indicating dependencies between properties. Technical report, Technische Universität München (2008)
16. Guizzardi, G.: Ontological foundations for structural conceptual models. PhD thesis, CTIT, Centre for Telematics and Information Technology, Enschede, The Netherlands (2005)
17. Aier, S., Riege, C., Winter, R.: Unternehmensarchitektur – literaturüberblick stand der praxis. Wirtschaftsinformatik 50(4), 292–304 (2008)
18. Schelp, J., Winter, R.: Language communities in enterprise architecture research. In: DESRIST 2009: Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, pp. 1–10. ACM, New York (2009)
19. Jonkers, H., van Burren, R., Arbab, F., de Boer, F., Bonsangue, M., Bosma, H., ter Doest, H., Groenewegen, L., Scholten, J., Hoppenbrouwers, S., Iacob, M.E., Janssen, W., Lankhorst, M., van Leeuwen, D., Proper, E., Stam, A., van der Torre, L., van Zanten, G.: Towards a language for coherent enterprise architecture descriptions. In: 7th International Enterprise Distributed Object Computing Conference (EDOC 2003), Brisbane, Australia. IEEE Computer Society, Los Alamitos (2003)
20. Jonkers, H., Goenewegen, L., Bonsangue, M., van Buuren, R.: A language for enterprise modelling. In: Lankhorst, M. (ed.) Enterprise Architecture at Work. Springer, Heidelberg (2005)
21. Österle, H., Winter, R., Hoening, F., Kurpjuweit, S., Osl, P.: Der St. Galler Ansatz des Business Engineering: Das Core Business Metamodel. Wisu – Das Wirtschaftsstudium 2(36), 191–194 (2007)
22. Kurpjuweit, S., Aier, S.: Ein allgemeiner Ansatz zur Ableitung von Abhängigkeitsanalysen auf Unternehmensarchitekturmodellen. In: 9. Internationale Tagung Wirtschaftsinformatik (WI 2007), Wien, Austria, Österreichische Computer Gesellschaft, pp. 129–138 (2007)
23. Le, L.S., Wegmann, A.: Definition of an object-oriented modeling language for enterprise architecture. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, HICSS 2005, p. 179c (2005)
24. Frank, U.: Multi-perspective enterprise modeling (memo) – conceptual framework and modeling languages. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS 2002), Washington, DC, USA, pp. 1258–1267 (2002)
25. Kirchner, L.: Eine Methode zur Unterstützung des IT-Managements im Rahmen der Unternehmensmodellierung. PhD thesis, Universität Duisburg-Essen, Berlin, Germany (2008)
26. Frank, U., Heise, D., Kattenstroth, H., Schauer, H.: Designing and utilising business indicator systems within enterprise models – outline of a method. In: Modellierung betrieblicher Informationssysteme (MobIS 2008) – Modellierung zwischen SOA und Compliance Management, November 27-28. Saarbrücken, Germany (2008)
27. OMG: Object constraint language (ocl) available specification, version 2.0 (formal/06-05-01) (2006)

28. Atkinson, C., Kühne, T.: Reducing accidental complexity in domain models. Software and Systems Modeling, 345–359 (2007)
29. Chair for Informatics 19 (sebis), Technische Universität München: Eam pattern catalog wiki (2010), http://eampc-wiki.systemcartography.info (cited 2010-02-25)
30. OMG: Meta object facility (mof) core specification, version 2.0 (formal/06-01-01) (2006)
31. Buckl, S., Franke, U., Holschke, O., Matthes, F., Schweda, C.M., Sommestad, T., Ullberg, J.: A pattern-based approach to quantitative enterprise architecture analysis. In: 15th Americas Conference on Information Systems (AMCIS), San Francisco, CA, USA (2009)
32. Johnson, P., Johansson, E., Sommestad, T., Ullberg, J.: A tool for enterprise architecture analysis. In: 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), Annapolis, Maryland, USA, October 15-19, pp. 142–156. IEEE Computer Society, Los Alamitos (2007)
33. Lankes, J., Schweda, C.M.: Using metrics to evaluate failure propagation and failure impacts in application landscapes. In: Multikonferenz Wirtschaftsinformatik, Berlin, Germany, GITO-Verlag (2008)
34. Fischer, R., Winter, R.: Ein hierarchischer, architekturbasierter ansatz zur unterstützung des it/business alignment. In: Oberweis, A., Weinhardt, C., Gimpel, H., Koschmider, A., Pankratius, V., Schnizler (eds.) Wirtschaftsinformatik 2007, Karlsruhe, Germany, pp. 163–180. Universitätsverlag Karlsruhe, Karlsruhe (2007)
35. Kurpjuweit, S.: Stakeholder-orientierte Modellierung und Analyse der Unternehmensarchitektur. PhD thesis, Universität St. Gallen (2009)
36. Niemann, K.D.: From Enterprise Architecture to IT Governance – Elements of Effective IT Management. Vieweg+Teubner, Wiesbaden (2006)
37. Yu, E., Strohmaier, M., Deng, X.: Exploring intentional modeling and analysis for enterprise architecture. In: Proceedings of the EDOC 2006 Conference Workshop on Trends in Enterprise Architecture Research (TEAR 2006), Hong Kong, p. 32. IEEE Computer Society Press, Los Alamitos (2006)
38. van Buuren, R., Jonkers, H., Maria-Eugenia, S.P.: Composition of relations in enterprise architecture models. In: The second International Conference on Graph Transformation (ICGT), Roma, Italy, pp. 39–53 (2004)
39. Englebert, V., Heymans, P.: Towards more extensible metacase tools. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 454–468. Springer, Heidelberg (2007)
40. Chair for Informatics 19 (sebis), Technische Universität München: Eam pattern catalog wiki (2009), http://eampc-wiki.systemcartography.info (cited 2010-02-25)
41. Ernst, A.: Enterprise architecture management patterns. In: PLoP 2008: Proceedings of the Pattern Languages of Programs Conference 2008, Nashville, USA (2008)
42. Noble, J.: Classifying relationships between object-oriented design patterns. In: Australian Software Engineering Conference (ASWEC), pp. 98–107. IEEE Computer Society, Los Alamitos (1998)
43. Kamlah, W., Lorenzen, P.: Logische Propädeutik: Vorschule des vernünftigen Redens, 3rd edn., Metzler, Stuttgart, Germany (1996)
44. Johnson, P., Nordström, L., Lagerström, R.: Formalizing analysis of enterprise architecture. In: Interoperability for Enterprise Software and Applications Conference, Bordeaux, France, p. 10. Springer, Heidelberg (2006)
45. Aier, S., Buckl, S., Franke, U., Gleichauf, B., Johnson, P., Närman, P., Schweda, C.M., Ullberg, J.: A survival analysis of application life spans based on enterprise architecture models. In: 3rd International Workshop on Enterprise Modelling and Information Systems Architectures, Ulm, Germany, pp. 141–154 (2009)

Playing ArchiMate Models

Jos Groenewegen[1], Stijn Hoppenbrouwers[1], and Erik Proper[1,2]

[1] Radboud University Nijmegen, Toernooiveld 135, Nijmegen, the Netherlands
[2] Public Research Centre Henri Tudor, 29 avenue John F. Kennedy
Luxembourg-Kirchberg, Luxembourg
jos.groenewegen@gmail.com, stijnh@cs.ru.nl, e.proper@acm.org

Abstract. This paper concerns the application of a gaming approach to the validation of ArchiMate models, with the aim of enhancing validation, by non-architects, beyond mere reading of the model. The game offers a guided process for systematic exploration of ArchiMate models, and for systematically raising questions about them. The development process and the design principles behind the game are discussed, as well as the information transformation involved in creating a model-specific game from an ArchiMate model. The game has been evaluated through application in a small real life case. We discuss the influence of our approach to model understanding by the players, and the conceptual merits and flaws of the game.

Keywords: enterprise architecture, architecture models, validation, games.

1 Introduction: Games for Enterprise Architecture?

This paper addresses the problem of validating the completeness and correctness of architecture models. It aims to do this by providing a proof of concept of a game (or 'game-like procedure') that helps validate architectures. We show that a game-based approach to architecture model validation has a fair amount of merit in helping to create a good understanding of an architecture model, a basic requirement for validation.

The concept of 'architecture' is broad and various definitions exist. We refrain from further discussion of the term here. ArchiMate models [1] are now the chosen architecture representation standard of The Open Group [2]; we assume this is enough reason to take them as a valid subject of our game. However, we do believe our approach could in principle be generalized to other types of (architecture) models. Exploration of such extension, however, is not within the scope of this paper.

Basic issues underlying our effort are what a game is, and what our specific requirements for a model validation game are. Definitions of "game" are as varied as the games that are out there [3]; as is the case with "architecture", no undisputable definition of "game" exists. Some literature is available on what makes games work [3,4,5,6]. An essential part of games, being interactive systems, is that there has to be some kind of interaction between at least one actor and the game; also, in multi player games, there is interaction between actors (players).

Games have a set of rules within boundaries of which the actors can operate and interact. These interactions should lead to fulfilment of some clear end goal. This can

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 182–194, 2010.

be both a goal in the game or, in the case of 'serious' games, a goal outside of the game (in our case, validation of a model: a 'utility goal' [7]).

In this paper, we follow [6, p15]: *"A game is a system in which players voluntarily engage in a goal-oriented, artificial conflict, that results in a quantifiable outcome. The activity takes the form of a process which is defined by rules, yet offers freedom of action."* This is a very workable definition in our case. Even validating an architecture model is a form of artificial conflict. The players need to overcome an artificial challenge: to provide structured information concerning the validity of the model, thereby increasing the chance that the model is actually valid. Within this scope a game can be developed that suits our utility goal.

Our game design is restricted by only a fairly rough and limited working set of requirements: besides aiding the validation of ArchiMate models, the game has to be learnable relatively quickly. Furthermore, the pre-existing knowledge required for playing the game needs to be fairly minimal.

2 Method

The main aim of our small project was showing the possible strength of a game based approach to validating ArchiMate models. We set out to create a playable game. The general methodological frame was design science [8]. We developed a prototype, tried it, and evaluated it, improving it as we went along. Fortunately, only a few cycles were required to create the final version presented here.

As discussed in [9], there is no existing set of comparable games. Thus, not only an objective measure of the game's quality was outside our reach, but so was even a simple comparison with another game. At the same time there is no good measure for the quality of architecture models [10,1]. Some work has been done on quality of modelling [11,12], but it cannot be readily applied to a game approach. The best we could do to show the game's value was to try it out and report on our experiences (and those of the players).

However, this does not mean we designed the game without basing it on some ideas and principles from the literature. Indeed, the first step was a literature study, which was undertaken in co-operation with the Netherlands Architecture Forum (NAF) Workgroup on Games and Architecture. [3] and [9] were taken as broad guidelines. After the literature study on existing games, the constructive part of the project began. The process of game creation was based on known methods for game construction [3,4,5], inevitably along with a substantial dose of creativity.

Testing a game is not trivial [3], especially in a field such as architecture. To achieve proof of concept we first applied the game in a "dry run" on the Archisurance case [13] to see if it fitted within the conceptual framework underlying ArchiMate. Next, we applied it in a small real life case to observe the effectiveness of the game. Interviews with game participants were held before and after the sessions to find out about the players' understanding of the architectural model and the correctness of the model. Although this by no means guarantees a perfect game it is a reasonable measure of the merit of the game concept, which enables us to find out if it warrants further research (in view of [14,15,16,17]).

3 Architecture Game Types

Before focussing on the design of a specific game, we addressed the general question of what types of architectural game could be usefully distinguished in the first place, based on their utility. This work was done in collaboration with a group of professional architects (plus one game design professional) at the NAF workgroup mentioned earlier. The categorization had to cover the full scope of how games could possibly be applied.

We identified the following four possible categories for games-for-architecting:

1. Convincing people of the added value of the concept of architecture through a game. An existing game in this vein would be the Ordina Alignment Game [14]. A simplified game simulation could work here, mirroring the problems architecture is supposed to remedy, and creating an experience as to how the remedy works.
2. Creating architecture: a game, or a set of integrated games, that support the creation of (representations of) architecture. No such games are known to us, nor is it clear what such a game might look like.
3. Analyzing and validating architecture representations. Again, no examples or existing ideas about how to go about creating such a game were found.
4. Creating awareness of a completed architecture among the stakeholders. Some examples are known [9]; a simulation game might also work here.
5. The categories have proven robust and seem to cover all relevant situations in the field.

As mentioned, some games of type 1 and 4 already exist, but not so for 2 and 3. Our research interest was thus more keenly raised by the latter types. The second category of game is rather ambitious and would need to support complex structured conceptualization. Though similar games have been considered for some other forms of modelling [18], for architecture we felt it would overstretch our current capacities. The third category of games, however, seemed promising for an initial attempt. It depends on an architecture (or architecture model) already being available, so the questions raised 'only' involve whether the architecture is a good one. The process to be followed can be quite different than one for the creation of an architecture. When an architecture is created one has to go through an elaborate creative process and generally come up with, and reach consensus on, 'new' knowledge. Evaluating a representation that has already been made seems much less taxing for what is, after all, only an initial attempt. Based on these considerations the choice was made to develop a game of the third type.

4 The Information Covered by ArchiMate Models

To find what information is typically expressed through ArchiMate models, we started by studying the ideas behind the ArchiMate language, as discussed in [13]. Several strict separations within the ArchiMate language are indicated. The first split is between the layers distinguished in the enterprise: the business layer, the application layer, and the technology layer. The three layers all reflect dynamic systems, that

can be framed in one meta-model. For more information on this meta-model, see [13], fig 7.

However, more information underlies ArchiMate models than can be captured by means of its meta-model. This concerns 'the stories behind the actors and objects'. These stories provide the reasons why, for example, actors are placed in the model. The idea behind ArchiMate models is that every concept should clearly contribute conceptually, both in the modelling language and in its application in an actual architecture model [10].

For dynamic systems, ArchiMate further splits the components into three categories: the active structure concepts, the behavioral concepts, and the passive structure concepts. A precise definition is the following, taken from [13]:

> *"Active structure concepts are concepts concerned with the execution of behaviour; e.g. (human) actors, software applications or devices that display actual behaviour. The behavioural concepts represent the actual behaviour, i.e. the processes and activities that are performed. The active structure concepts can be assigned to behavioural concepts, to show who (or what) performs the behaviour. The passive structure concepts are the concepts upon which behaviour is performed."*[13, page 9]

In other words: *who* does things, *what* do they do, and *on what* do they do them [10,13]. A clear relation exists with basic semantic roles in natural language sentences (subject, predicate, object).

A further split in ArchiMate concerns the three kinds of components on the one hand (active structure, behaviour, passive structure), and the *relations* between these components on the other.

We observe that what is missing is some *context-related information*. As mentioned, the split into three components is based on the structure of natural language and the way we reason about active processes [13]. As discussed in [15,17,16], more information is needed for this. The 'who'-'what'-'on-what' split that ArchiMate makes is often seen in representations of action. However, to have a full understanding of action, a fourth parameter has to be added: the *why* of the action. There is some reason behind the action being undertaken; no action is taken for sake of itself. If there is a reason for existence of every part (concept) in an architectural model, then the architect should have thought about this at least in principle. And yet, such information is not captured through the meta-model. Thus, the validation of ArchiMate models should benefit at a fairly fundamental level from taking such information into account.

As a foundation of the validation game we will use the three types of component: active structure elements, behavioural elements and passive elements. Because they are also basic concepts in natural language, humans can understand these fairly well [13,15,16,17]. For full understanding, however, we will need a fourth type of component: the reasons underlying action/behaviour as modelled. We work under the assumption that the model (explicit knowledge) and the architect's tacit knowledge about the choices made are somehow available to players of the game, if only after questioning. With this knowledge we seem to have everything we need to validate both the architecture model and (some of the) rationales behind it [2].

5 A Game for Architecture Model Validation

In this section, which is the core of the paper, we describe the game design and its underlying principles.

5.1 Aim of the Game

It is clear that a formal validation of an ArchiMate model should not be the aim. We are primarily concerned with a qualitative, human-knowledge-based type of validation, with a certain degree of subjectivity. This is inherent in all stakeholder-based validation. The second aim of the game is accessibility. The whole idea of a game to aid validation is that it does not typically involve the creators of the model (architects), but stakeholders with no special modelling skills.

Please note that the game is merely a proof of concept. It does not need to be optimized in terms of effort needed to prepare, or actual execution of the game. It has to be good enough to show that the approach is viable.

5.2 Basic Design Choices

The first design choice is that we want to make sure that all relevant knowledge is available through the players involved. No constraints are put on who can play, as long as all knowledge required is covered within the set of players.

The second choice is that we will base each instance of the game on a specific, existing ArchiMate model. This means the relation between the model and the game should be clear at all times. Whatever final form the game takes, we should be able to trace it back to the original ArchiMate model in such a way that the model can directly benefit from the game.

Next, the reasons behind the interactions between the elements of the ArchiMate model have to be incorporated (as explained in section 4).

5.3 The Game

Now that the core principles of the design are clear we arrive at a major creative step. As discussed, a main lack of knowledge required for validation lies in the implicitness of knowledge of why the architect puts forward “actors acting on elements”. Such knowledge is largely tacit, process-related, and hard to make explicit. It involves an intuition (of the people doing the work) of how things work and how the process flows, which the architect has to distil a model of [19]. The first step in the creative process was the realization that dealing with intuition is a major problem in traditional model validation, which could be better tackled through a game approach allowing for a not too strictly regimented process. This should lead to better understanding of the architect’s intent and rationales.

Active components display behaviour and they do this on passive structure elements. The possible behaviour that is displayed, and by whom and on whom, is all captured in the architecture model. If we create the possibility of “walking through” the architecture in some sort of logical order, a client can become an actor on another actor, which implies a temporal dependency.

From here we went to the insight that every interaction captured in an architecture model entails a similar dependency: an actor acts in a certain way on a client, for a reason. In ArchiMate, such interactions are always depicted by means of minimally two elements. In other words, they are never 'hidden' in one element. Because of this it is possible to see every action taken, and in this way step through the model. Thus, one can incorporate a temporal aspect into the relations.

Every active structure element has a starting point in the architecture, a 'path of influence' through the architecture, and an ending point where the original actor ends up. The path the actor takes through the architecture, and the actions it takes, can thus be seen, followed, and understood. This concept lies at the core of the game. The players need to be forced to think actively about the behaviour the actors display. The chosen way to do this is to have the players play the actors associated with their specific domain expertise. Every actor has a starting point and an end point in the architecture (not necessarily different ones), and paths may be shared. The players' choices at every possible step represent the behaviour of the actor, and should coincide with the behaviour the architect had in mind when creating the model.

In the current context, we assume that if an architect has a correct idea of the actual situation in the domain, the ensuing model is produced correctly. In other words, we assume (naively) that architects are fluent and flawless in ArchiMate and are also perfect in performing the preparation. Admittedly, the architect may influence the outcome of the game though the way the game is set up by him. Also, one may wonder whether third-party validation, divorcing the architect form the process, would be preferable. However, cutting out the architect form the validation process would require for someone else to do the game preparation. Since this requires in-depth knowledge 'behind the model', and since automated game generation is not an option at this stage, we cannot but acknowledge these points but have to simply assume they do not arise.

It is time to take a look at what all this leads to. After presenting the game rules we will discuss how an ArchiMate model can be transformed into a playable game.

5.4 Basic (generic) Game Instructions

We will now discuss how the game is set up in general, excluding mode-specific aspects. For a discussion of the preparations for this phase see 5.5.

Setting up the game board
Place different coloured pawns (or marked pawns) at each starting location marked by the game master (typically, the Architect). Each marked start location has one or more designated end locations. The map is laid out, meaning the assignment points are laid down as well as the routes between them (which are still locked).

Start
Each player moves in turn, starting with the youngest player. A player goes through the following steps (some of which are not always performed, depending on the situation):

1. Move a pawn to any 'reachable' assignment, and try to complete the assignment.
2. Pick up available items on routes or in squares, to assist with or as required for assignments.
3. Trade items with another pawn it can reach.

4. Drop available items on a tile or on a route it can reach, for other pawns to pick up.

A player can go through steps 2 to 4 as often as he likes. Step 1 may be performed at most 3 times in a row before another player gets a turn (who might skip it if he wants).

Game goal
Pawns start at their designated start location and can move across any explored path as far as they want. They have one or more designated end points. When an unexplored path is entered, the pawn has to 'open' the associated card (provided by the game master) and complete the assignment in order to continue. If the pawn withdraws after reading the assignment, the path stays closed. Opened paths can generate passive structure elements in their own or other tiles (as described on the card). Once a pawn reaches an end point, then its route (every step in which it undertook action) is 'marked' and a new pawn is generated at the start location.

The game ends when a line can be drawn, across marked squares and routes, between all end locations.

5.5 Preparations by the Game Master

The basic instructions are simple but the preparation for an actual game is more complex. Within the model to be evaluated, all active structure elements need to be marked. Any active structure element that comes from an earlier 'interaction', or involves an actor of an earlier interaction, is marked as an assignment point. If there are no further routes to take for an actor from an assignment point, it is also marked as an end location. Any active structure element with no previous location is marked as a starting location. The passive structure elements are incorporated into the game as items that can be carried along. The behavioural elements are the choices that force people down a route. Furthermore, they can be 'nodes' where several routes cross. A node can be the creator of passive structure elements, an assignment, or merely a crossroad.

To help clarify this, we provide an example game transformation for the archisurance example. The archisurance model shown in fig. 1 is taken from [2].

Within the model we first need to mark the components. Thin squares lines designate active structure elements. A thicker line marks start or end points, depending on ingoing and outgoing lines. Circles mean behavioural elements. A cross is a passive structure element. Some liberty has been taken with respect to the intent and meaning behind this model, for explanatory reasons. Thus we arrive at the marked model shown in fig 2.

For the sake of workability in this example we will not add all the in-depth options further expanding the model. The marked model given is still incomplete. Where is the architect's tacit knowledge about the process flow?

The preparations to make this explicit can be quite extensive as the implicit knowledge is written down in game form through requirements for opening up nodes and possible transformations. Let us see what it would look like when filled in for the given archisurance example. First we will give the options at each node or point of behaviour. Then we will make explicit demands for any routes that exist.

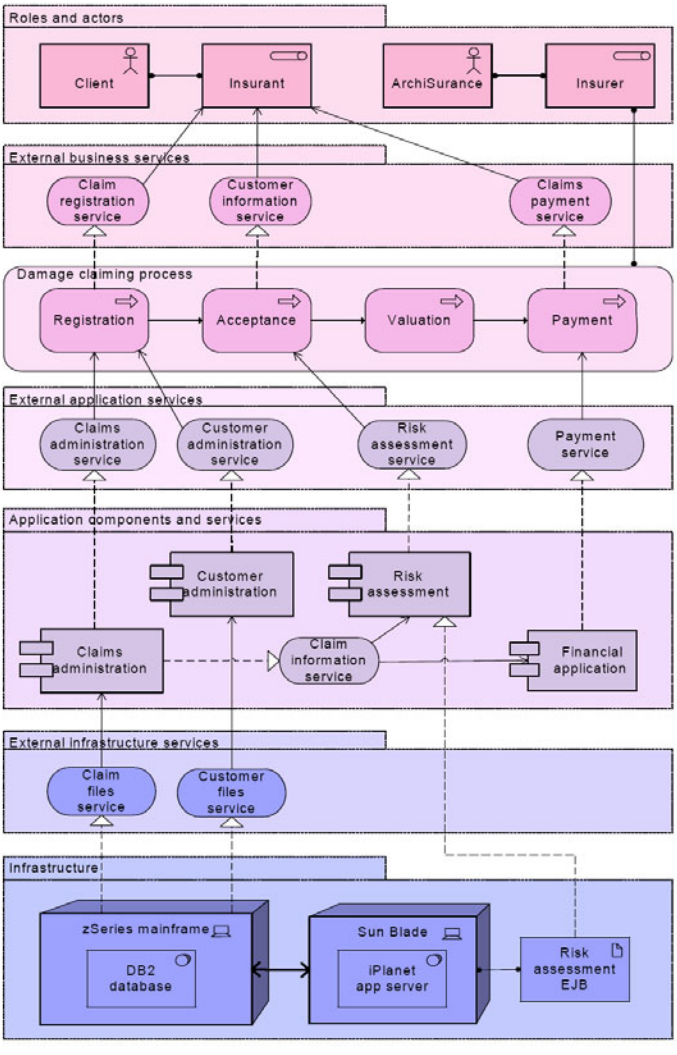


Fig. 1. A targeted ArchiMate model

1. Client/Insurant
2. Registration a. With claim registration service and claims administration service and customer administration service create filled claim registration service
3. Acceptance a. With customer information service and risk assessment service create filled customer information service
4. Valuation a. Create cost item
5. Payment a. With cost item, payment service and claims payment service create filled claims payment service

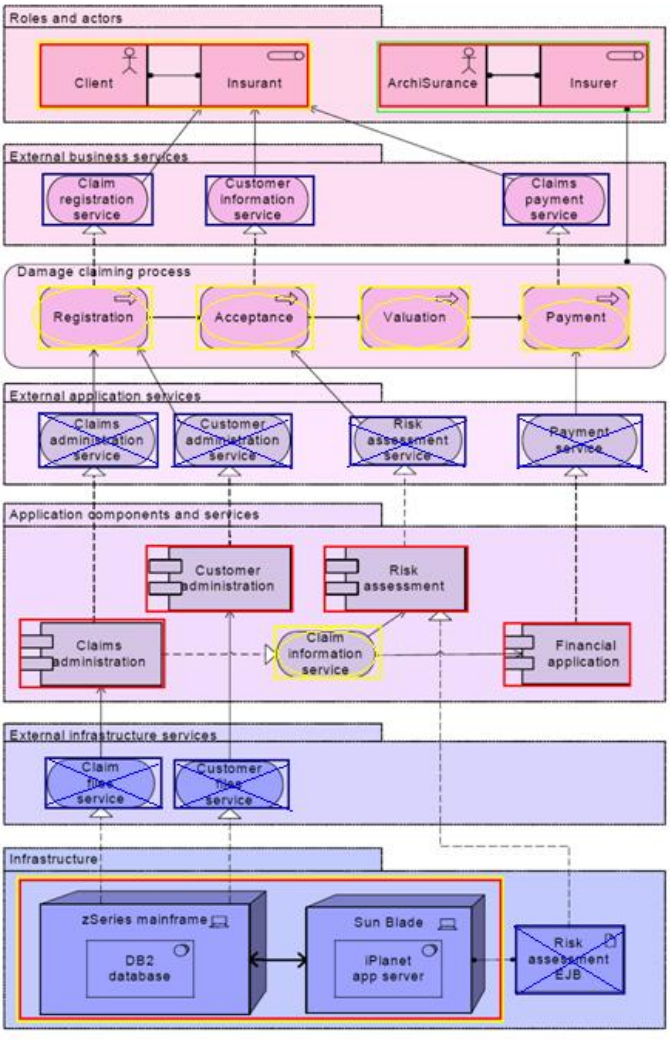


Fig. 2. An ArchiMate model marked as a "game board"

6. Insurer a. Be end point
7. Claims administration a. With filled claim registration service create fulfilled claims administration service
8. Customer administration a. With filled claim registration service create fulfilled customer administration service
9. Risk assessment a. With filled customer information service and risk assessment service and risk assessment EJB create fulfilled risk assessment service
10. Claim information service a. Choose to go to risk assessment or financial applications

11. Financial administration a. With payment service and filled claims payment service create fulfilled payment service
12. DB2 Database

Route demands (if none stated none exist):

1. Registration -> acceptance requires filled claim registration service
2. Acceptance -> valuation requires filled customer information service
3. Valuation -> payment requires cost item
4. Payment -> insurer requires filled claims payment service && if player = brown -> fulfilled payment service
5. Claims administration -> claims information service requires fulfilled claims administration service
6. DB2 database -> Risk assessment requires claims administration -> claims information service = open

The somewhat intimidating amount of information here reflects the amount of information contained in architecture models. The important advantage of our approach is that this information is given to players in an explicit format in the form of cards and clear goals. The abstract layers and width of the model are replaced by understandable goals and paths.

6 Evaluation

A 'real-life' example model was played and evaluated using an architecture model of the small organisation the players were part of.

6.1 Main Findings

What went well?
The preparations were found to be time consuming, but no inherent difficulties were encountered going through this step. As the game itself unfolded, results were promising. Through the game, the staff members participating indeed developed a better understanding / feel for the model than they had with the original model, and the sense of model complexity was reduced.

Points of improvement
When players were found to miss out on information from the model it was difficult to note the exact knowledge gap. Although the area of missing knowledge was easy to find, the precise knowledge gap could not be identified. In other words: seeing that a model is not valid is easy, but pinpointing why the model is not valid is more difficult.

Lessons learned
Not unexpectedly, we found that it is very important to ensure the knowledge of the players fits the domain they are playing. If knowledge is missing flaws in the model can be missed.

A second lesson learned concerns the 'after-game' phase. To maximize understanding of what the players noted, and what they went through, debriefing the players after

the game is important. Although doubts about the model should in fact be expressed during the game, people tend not to do so, especially in a social context with colleagues present. Furthermore, debriefing helps consolidate the concepts of the architecture, and the understanding a player has of her own work processes.

6.2 Model Understanding by the Players

Through playing the game the players achieved a considerably better understanding of the contents of the model than by merely looking at the model. They were far more capable to see the relations between elements after they actually experienced them. Furthermore, due to the explicit and mandatory constraints between moves (see "game preparations"), the actual relations fitted better with their expectations and their experience of them in reality. However, no new knowledge of the actual organisation was attained, while some organisational knowledge was felt to be missing for certain people. Post-game evaluation remedied this, as seven interviews before and after the game were included in the total setup.

6.3 Investments Required

The initial investment to get the architecture into a 'game' mode is considerable. It requires making tacit knowledge explicit and writing down what the ideas behind all parts of the architecture are. However, it does not seem unreasonable that this should in some way be part of routine architecture documentation. We estimate that the transformation takes roughly same amount of time as the creation / writing down of the architectural model (assuming the knowledge required is available).

The investment of playing the actual game is estimated to range up from 30 minutes for small models to several hours for larger models, especially for those that require a significant increase in player numbers to have all the necessary knowledge available in the game.

The time needed to process the results varies depending on the results. If no flaws are found the results can be processed in several minutes, plus evaluation time with players. However, if flaws are found in the model it can take a significant time investment to pinpoint the exact flaw and correct it.

7 Conclusion

Let us look back at our original goal: "To provide a proof of concept for a game- based approach to validating ArchiMate models." Indeed, a game-based approach to validating ArchiMate models can work. It does not give 100% certainty the model is valid, but it seems to provide real added value. It introduces a new and different, systematic way of validating a model. It has different requirements from most methods, most notably the need to make explicit tacit knowledge in a multi-dimensional space, based on location, internal relations, and time. Preparing the game could in fact be seen as an additional highly useful activity in model validation. The game may well open up model validation to people who before could not be involved. So, the approach has merit. Further research will be necessary to show the extent of this merit in more complex cases, and more extensive real-life situations. In addition, some further study

concerning the detailed interactions taking place as the game proceeds could render good insights as well as input for improvement of the game's playability [7].

References

[1] Lankhorst, M.M. (ed.): Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer, Berlin (2005)

[2] Iacob, M.-E., Jonkers, H., Lankhorst, M.M., Proper, H.A.: ArchiMate 1.0 Specification. The Open Group (2009)

[3] de Caluwé, L., Hofstede, G.J., Peters, V.: Why do Games Work? Kluwer, Deventer (2008)

[4] Hunicke, R., LeBlanc, M., Zubek, R.: MDA: A Formal Approach to Game Design and Game Research. In: Proceedings of the AAAI 2004 Workshop on Challenges in Game AI, pp. 1–5 (2004)

[5] Salen, K., Zimmerman, E.: Rules of Play, Game Design Fundamentals. MIT Press, Cambridge (2004)

[6] Wilmont, I.: A gaming approach to collaborative modelling. Master Thesis in Information Science, Radboud Universiteit Nijmegen, Thesis Number 91-IK (2009)

[7] Hoppenbrouwers, S.J.B.A., Weigand, H., Rouwette, E.A.J.A.: Setting Rules of Play for Collaborative Modelling. In: Rittgen, P. (ed.) International Journal of e-Collaboration, vol. 5(4), pp. 37–52. IGI Publishing, USA (2009), Special Issue on Collaborative Business Information System Development

[8] Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. MIS Quarterly 28, 75–106 (2004)

[9] van der Bij, H.: Lexicon Architectuurspellen. In: Landelijk Architectuur Congres 2009 (November 27, 2009) (in Dutch)

[10] Proper, H., Verrijn–Stuart, A., Hoppenbrouwers, S.: Towards Utility–based Selection of Architecture–Modelling Concepts. In: Hartmann, S., Stumptner, M. (eds.) Proceedings of the Second Asia–Pacific Conference on Conceptual Modelling (APCCM 2005), Sydney, New South Wales, Australia. Conferences in Research and Practice in Information Technology Series, vol. 42, pp. 25–36. Australian Computer Society (2005)

[11] van Bommel, P., Hoppenbrouwers, S.J.B.A(S.), Proper, H.A(E.), Roelofs, J(J.): Concepts and Strategies for Quality of Modelling. In: Halpin, T., Krogstie, J., Proper, E. (eds.) Innovations in Information Systems Modelling, Methods and Best Practices. Advances in Database Research series, ch. IX, pp. 167–189. IGI Global Publishing, USA (2008)

[12] Krogstie, J., Sindre, G., Jorgensen, H.: Process models representing knowledge for action: a revised quality framework. European Journal of Information Systems 15, 91–102 (2006)

[13] Lankhorst, M., Proper, H., Jonkers, H.: The Architecture of the ArchiMate Language. Enterprise. In: Proceedings of the EMMSAD 2009, held at CAiSE 2009, Amsterdam, the Netherlands. LNBIP, vol. 29, p. 367. Springer, Berlin (2009)

[14] Ordina-allignment game, `http://www.ordina.nl/Downloadcentrum/~/media/Files/Onze%20dienstverlening/Consulting/The%20Alignment%20Game.ashx?forcedownload=1`

[15] Taylor, J.: The "rational" organization reconsidered: An explanation of some of the organizational implications of self-organizing. Communication Theory 11(2), 137–177 (2001)

[16] Drew, P., Heritage, J.: Talk at work. Cambridge University Press, Cambridge (1992)
[17] Garcia-Lorenzo, L., Nolas, S.-M., de Zeeuw, G.: Telling stories and the practice of collaboration. International Journal of Sociology and Social Policy 28(1–2), 9–19 (2008)
[18] Hoppenbrouwers, S.J.B.A., Schotten, B.: A Game Prototype for Basic Process Model Elicitation. In: Persson, A., Stirna, J. (eds.) The Practice of Enterprise Modeling, 2nd IFIP WG8.1 Working Conference, PoEM 2009. LNBIP, vol. 39. Springer, Heidelberg (2009)
[19] Zachman, J.A.: A framework for information systems architecture. IBM Systems Journal 26(3), 276–292 (1987)

Supporting Layered Architecture Specifications: A Domain Modeling Approach

Jenny Abramov and Arnon Sturm

Department of Information Systems Engineering,
and Deutsche Telekom Laboratories
Ben-Gurion University of the Negev
Beer Sheva 84105, Israel
{jennyab,sturm}@bgu.ac.il

Abstract. Software architectural patterns help manage complexity through abstraction and separation of concerns. The most commonly used architectural patterns are layered architectures, which benefit from modularity and reuse of layers. However, they lack in supporting changes, as there is a need to do a substantial amount of rework on the layers in order to incorporate changes. Furthermore, the comprehension of specifications which are based on a layered architecture can be difficult. In order to address the aforementioned limitations, we adopt a domain engineering approach called Application-based Domain Modeling (ADOM). Using ADOM, we refer to each layer as a separate domain model, whose elements are used to classify the application model elements. Consequently, the application model is represented in a unified form, which incorporates information from all of the layers. This allows performing changes in the model, without creating cascades of changes among the layers' models in order to synchronize them.

Keywords: Layered architecture, ADOM, UML, Domain modeling.

1 Introduction

As software systems are complex, they must be built on a solid foundation, namely their architectural design. A major mean for designing software architectures is architectural patterns, which are used to enhance modularity by helping in managing complexity through abstraction and separation of concerns. Architectural patterns specify the structure of a system by providing a set of predefined subsystems, specify their responsibilities, and include rules and guidelines for organizing the relationships between them [3].

Among the various software architectural patterns the most common one is the layered architecture [5]. The notion of layered systems has become popular since it was first introduced at 1968 by Dijkstra for the "THE system" [4] operating system. It was also presented in the OSI seven layer network architecture [18] and in the area of artificial intelligence [2]. A key principle of a layered architecture [5] is to factor out responsibilities into separate cohesive units and define the dependencies between them. Layers are meant to be loosely coupled, with dependencies in only one direction such that the

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 195–207, 2010.

lower layers provide low-level general services, and the higher layers are more application specific. Collaboration and coupling is from higher to lower layers. Thus, the layered architectural style introduces a hierarchical subsystem structure. The hierarchical structure considered as a good structure to deal with complex systems in a scalable way. Moreover, other architectural styles, as the component-based architecture, arrange their concepts in layers [17]. The functionality and the amount of layers vary across applications. However, there are some fairly standard layers. For instance, the typical architecture style for user-oriented systems is the standard three layers architecture, which includes the presentation, the business logic, and the persistence layers.

A layered architecture can help mitigate complexity and improve understandability by grouping functionality into different areas of concerns. It supports design based on increasing levels of abstraction, which enables the designer to partition a complex problem; and it improves the maintainability and extensibility of the application by minimizing dependencies, allowing exchange of layers, and isolating technology upgrades. However, the standard layered architectures lack in supporting changes that have to be propagated through multiple layers, since those changes cause a cascade of changes on many layers in order to incorporate apparently local changes. For example, in many user-oriented systems, such as information systems, many entities require representation in several layers. In that case, a change in the type of an attribute will require from the developer to find all the related entities in all layers in order to change the type of the corresponding attributes. Since software design generally evolves over time, it is important to find a way to overcome those limitations. We believe that the aforementioned reasons also hinder the comprehension and the ease of construction of large and complex systems.

Software systems are usually specified via models. There are two major approaches for utilizing architectural patterns though models: Architecture Description Languages (ADLs) which aim at formally representing software architectures and the Unified Modeling Language (UML) which is a generic modeling language that can also be used to describe software architectures. These approaches provide methods and tools for representing and analyzing architectural designs; however, since these are general-purpose approaches, it is difficult to address the aforementioned specific limitations of layered-based applications. Motivated by the popularity of layered architectures and the limitations inherent to general-purpose approaches, this paper presents an idea of a modeling approach specifically for modeling layered architectures.

In this paper we adopt a domain engineering approach called Application-Based Domain Modeling (ADOM), which enables specifying and modeling domain artifacts that capture the common knowledge and the allowed variability in specific areas, guiding the development of particular applications in the domain, and verifying the correctness and completeness of applications with respect to their relevant domains. Referring to the layered architecture, each layer is represented as a domain model and application model elements are classified by the layers' (domain) models elements. In that case, the designer needs to manage a unified application model without losing the architectural information. Thus, when dealing with a change, the designer makes the required change on the unified model without being required taking care of cascading changes among layers. We term the resulted model a Multi-Classified Model (MCM).

The structure of this paper is as follows. Related work is presented in Section 2. Section 3 introduces the ADOM approach whereas Section 4 elaborates on the proposed

approach, the Multi-Classified Model (MCM). Section 5 summarizes the strengths of the MCM and finally, Section 6 concludes and discusses future plans.

2 Related Work

As software architecture and the practice of architectural design have been recognized as significant issues in complex software systems engineering. Researchers and practitioners have been proposing numerous[1] formal notations for representing and analyzing architectural designs. These methods were meant to increase comprehension of architectural designs, to improve the ability to analyze consistency and completeness, and to support changes of the design during the development lifecycle. Architectural design approaches can be classified into three research areas: (1) ADL-related approaches, (2) UML-based approaches, and (3) the combined approaches.

Most ADLs support formal architecture representation including topological constraints from a structural point of view. They vary in the level of abstraction they support, their terminology, the information they specify and the analysis capabilities they provide. Each ADL provides certain distinct capabilities. Most first generation ADLs were developed to support some specific software architecture aspects. For instance, Rapide [6] is a language for modeling architectures of distributed systems. It is an event-based architecture definition language. Wright et al. [19] formalize the semantics of architectural connections. Darwin [8] is designed for dynamic architecture using π-calculus to formalize architectural semantics. Second generation ADLs identify and process fundamental concepts common to first generation ADLs in order to allow architectural interchange. For example, ACME [6] was developed to provide a framework to integrate different ADLs by supporting mapping of architectural specifications from one ADL to another.

Since ADL users were required to learn the specific notation of each ADL, and ADLs were not integrated in any development process, ADLs have not come into extensive use in the industry. Hence, the usage of a standard language such as UML for describing architectural design might make it easier to understand, mitigate the effort of preserving the architecture consistent during development phases, and it would be supported by existing and fairly standard tools. Furthermore, as UML has become standard general modeling language for software development, representing the architecture with UML will allow integrating it with the rest of software artifacts.

However, UML is not designed, syntactically or semantically, to represent software architecture elements and therefore does not support some architecture concepts, such as the lack of supporting connectors and architectural styles [19]. For that reason, many studies suggested extending the vocabulary and semantics of UML to apply its modeling capabilities to the concepts of architecture design. It is important to mention that UML 2.0 embraces much more constructs that are important to architecture description than UML1.x, such as components and connectors. Medvidovic et al. [10] identified three approaches for modeling software architectures using UML: (1) using UML "as is" [8]; (2) Extending UML in a heavyweight way [13] by adding new modeling elements or replacing existing semantics via direct modification of the

[1] According to Malavolta et al. [9] there are more than 50 ADLs proposed in academia and industry.

UML metamodel; and (3) Extending UML in a lightweight way [10][20] by using the extension mechanism of UML for defining new modeling elements. The adaptations are defined using stereotypes, which are grouped in a profile.

In addition to ADLs and UML approaches there are some combined approaches. These approaches seek to combine ADLs with UML notations. For example, Roh et al. [16] suggest a layered language which uses UML, generic ADL comprising of domain-independent elements, and a domain-specific ADL.

There are some important architecture specific problems that can be minimized or avoided by concentrating on a specific architectural style, such as the layered architectures. However, all of the approaches mentioned above do not deal with the disadvantages of some specific architectural style but concentrate on a general representation of software architectures and therefore cannot take advantage of the unique properties of some specific architectural style.

In this paper we introduce our approach to support structural layered architectures' specification. We use as an example the standard three layered architecture, which is the most basic and common structure in user-oriented systems [1][12].

3 Application-Based Domain Modeling (ADOM)

The Application-based Domain Modeling (ADOM) [14][15] is rooted in the domain engineering discipline, which is concerned with building reusable assets on one hand and representing and managing knowledge in specific domains on the other hand. ADOM supports the representation of reference (domain) models, construction of enterprise-specific models, and validation of the enterprise-specific models against the relevant reference models. The architecture of ADOM is based on three layers:

(1) The language layer comprised of metamodels and specifications of the modeling languages. In this paper we use UML 2.0 class diagrams as the modeling language, since the focus of this paper is on the structural aspect of the architecture.

(2) The domain layer holds the building elements of the domain and the relations among them. It consists of specifications of various domains; these specifications capture the knowledge gained in specific domains in the form of concepts, features, and constraints that express the commonality and the variability allowed among applications in the domain. The structure and the behavior of the domain layer are modeled using a modeling language that is defined in the language layer. In this paper we refer to the structure of each layer as a domain model.

(3) The application layer consists of domain-specific applications, including their structure and behavior. The application layer is modeled using the knowledge and constraints presented in the domain layer and the modeling constructs specified in the language layer. An application model uses a domain model as a validation template. All the static and dynamic constraints enforced by the domain model should be applied in any application model of that domain. In order to achieve this goal, any element in the application model is classified according to the elements declared in the domain model using UML built-in stereotype and tagged values mechanisms. In this paper the application model elements are multi-classified by the layers' (domain) model elements.

For describing variability and commonality, ADOM uses multiplicity stereotypes which can be associated to all UML elements, including classes, attributes, methods, associations, and more. The multiplicity stereotypes in the domain model aim to represent how many times a model element of this type can appear in an application model. This stereotype has two associated tagged values, min and max, which define the lowest and the upper most multiplicity boundaries, respectively. For clarity purposes, four commonly used multiplicity groups were defined: <<optional many>>, <<optional single>>, <<mandatory many>>, and <<mandatory single>>. The relations between a generic (domain) element and its specific (application) counterparts are maintained by the UML stereotypes mechanism: each one of the elements that appears in the domain model can serve as a stereotype of an application element of the same type (e.g., a class that appears in a domain model may serve as a classifier of classes in an application model). The application elements are required to fulfill the structural and behavioral constraints introduced by their classifiers in the domain model. Some optional generic elements may be omitted and may not be included in the application model and some new specific elements may be inserted to the specific application model, these are termed application-specific elements and are not stereotyped in the application model.

ADOM also provides a powerful verification mechanism that prevents application developers from violating domain constraints while (re)using the domain artifacts in the context of a particular application. This mechanism also handles application-specific elements that can be added in various places in the application model in order to fulfill particular application requirements.

4 The Multi-Classified Model Approach

In this paper we propose the Multi-Classified Model (MCM) approach for specifying layer-architecture-based applications. Working with MCM advocates the following steps: (1) Modeling a layered application; (2) Verifying the application against the predefined layers (as domains); and (3) Transforming the model into a layered application. In this paper, we focus on the first step and partially on the second step. In this paper, the MCM approach uses ADOM [14][15] with UML 2.0 class diagrams as the modeling language in order to describe the static structure of a system in terms of classes and their relationships.

As this paper focuses on the application designer activities, we assume that a software architect has already provided the following artifacts: (1) a set of domain models that represent the layers; (2) the dependency among the layers (domains); and (3) the dominant domain, which is used for creating the initial skeleton of the application, as it represents the core of the application domain.

Having the domain and architectural knowledge, the designer should preserve the following steps while specifying the (layered) application model:

1. Automatically generate an initial application model based on the dominant domain model, as common in ADOM.
2. Manually modify and refine the application model, based on the application specifications.

3. Manually classify the model according to the various layer (domain) models, which were provided by the software architect.
4. Automatically verify the application model with respect to the layer (domain) models. This is done using the ADOM validation procedure for each model separately.

Note that the various steps can be done iteratively.

In order to demonstrate the MCM approach we use a *ticket selling system* designed as a standard three layer architecture application. The tickets selling system allows registered users to buy and sell tickets to events. As a modeling infrastructure, for the *ticket selling system*, a designer has three domain models representing the three layers[2]: (1) the Web Interface (WI) domain, which represents the presentation layer, is depicted in Fig. 1; (2) the Registration (R) domain, which represents the business logic layer, is depicted in Fig. 2; and (3) the Relational Database (RD) domain, which represents the persistent layer, is depicted in Fig. 3. The dependencies among the domains are defined as follows: WI depends on R and R depends on RD, which correspond to three layered architecture. The dominant domain is the Registration (R) domain since it is the core asset for applications in this domain.

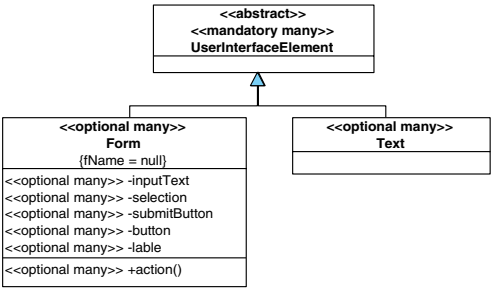


Fig. 1. The Web interface domain model

The domain models presented in Fig. 1, Fig. 2. and Fig. 3. state that all application models that are derived from them should follow the rules (among others) of:

- At least one *Form* or *Text* page should be specified in the application model, as specified within the *Web Interface* (WI) domain in Fig. 1.
- At least one *Client*, one *Provider*, one *Product*, one *ProductItem*, and one *Registration*, should be specified in the application model while preserving the structure as specified within the *Registration* (R) domain in Fig. 2.
- At least one database *Table* should be specified in the application model as specified within the *Relational Database* (RD) domain in Fig. 3.

In addition, the applications that follow these domains should specify the attributes and the operations of each class following the specifications of the domains. For example, each application has to have at least one *ProductItem*, which exhibits zero or more *detail* attribute, possibly one Boolean operation *checkAvailability* of the product, possibly

[2] Note that these three domains models are partial and we kept them small in size for the purpose of presenting the MCM-based approach principles.

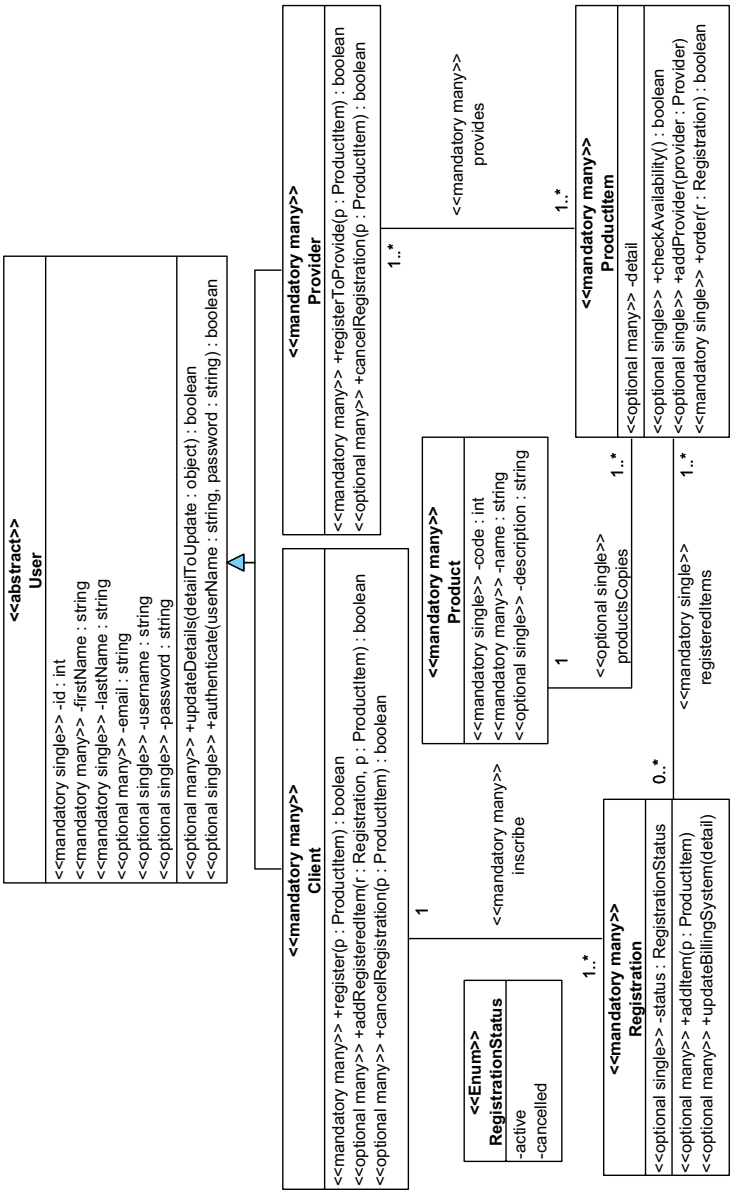


Fig. 2. The registration system domain model

one *addProvider* operation, and exactly one Boolean *order* operation. As common in ADOM, the relations between a layer (domain) element and its application counterparts are maintained by UML stereotypes, such that a domain element serves as a stereotype of an application element.

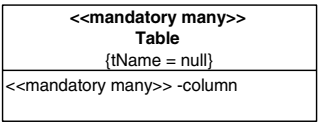


Fig. 3. The relational database domain model

Since there is a many-to-many relationship between the application elements in the various layers, tagged values associated with the domain elements stereotype can be used as parameters in the application model. For example, in the *Relational Database* (RD) domain, in Fig. 3, the *tName* tagged value associated with *Table* allows to specify in the application model the table name, to which the application element is associated. Tagged values associated with attributes or operations in the domain model are not presented due to visibility reasons. For instance in the *Relational Database* (RD) domain, in Fig. 3, a *key* can be associated with table *column*, this tagged value can be used to indicate the key type of a column in the database table.

Having domain models representing the various layers and the modeling infrastructure, the designer can generate the initial application model, modify and refines it, and finally classify it following the guidelines provided by the layer (domain) models. As in all layered architectures, each application element is associated with at least one layer (domain). For each such layer, an application element can either be classified by a specific layer (domain) element, or otherwise, be an application-specific element in that layer, and to be classified by the layer (domain) itself.

Fig. 4 depicts the resultant model of the *ticket selling system*. The model, in Fig. 4, also shows that this application is structured by the three layered architecture which is defined by three domains: Web Interface (WI), Registration (R), and Relational Database (RD). Each class is classified according to its requirements to the appropriate layers defined by the relevant domain models. From this representation we can easily understand the classification of each element with respect to the layers it belongs to. Any element which is application-specific is classified by the domain, but not by domain elements. In Fig. 4, the *Seller*, whose details are generalized from *User*, is classified as *<<WI.Form fName=login>>*, *<<WI.Form fName=myAccount>>*, *<<R.Provider>>*, *<<RD.Table tName=sellers>>*, and *<<RD.Table tName=orders>>*. This means that the *Seller* class has impact on all layers, the *Seller* class is represented in two forms: *login* and *myAccount*, it is follows the *Provider* class from the *Registration* (*R*) domain and maintains persistent data in two database tables: *sellers* and *orders*.

The attributes and the operations of each class are also classified according to the attributes and operations of the assigned domain classes, i.e. the *id* attribute in the *User* class is classified *<<User.id>>*. Similar to class classification, the attributes and operations that are application-specific elements are classified by the layer (domain) name they belong to. For example, the *orderNum* attribute in the *Order* class is classified *<<R>>* meaning that it is an application specific element that belongs to the registration layer.

For clarity purposes, we recommend presenting only the attributes and operations classification of the dominant domain layer, in the case of the *ticket selling system* it is the *Registration* (i.e., the business logic) layer.

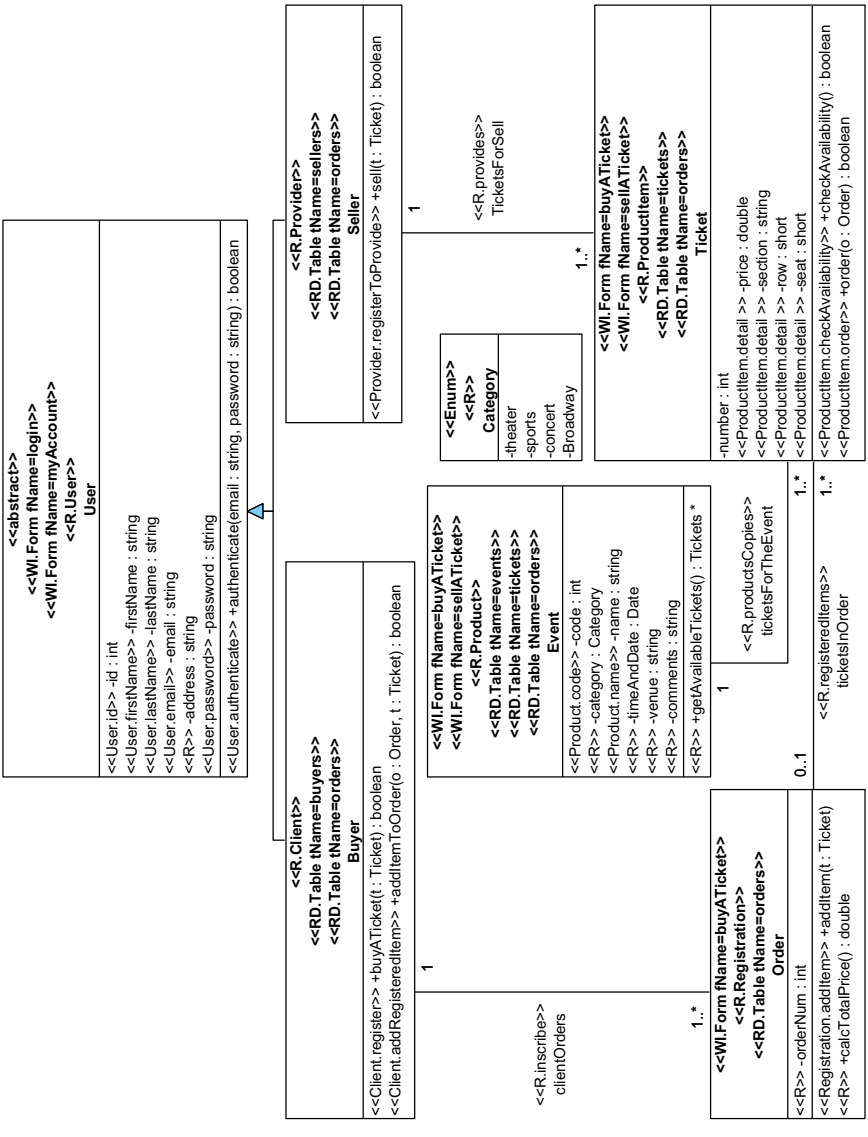


Fig. 4. The MCM tickets selling system application model

Fig. 5 shows the hidden classifications of the *Ticket* class. It specifies that the *number* attribute is a column in the *tickets* and *orders* tables and it also serves as a primary key in the *tickets* table. In addition, the *price*, *section*, *row*, and *seat* attributes are columns in the *tickets* table and they are presented to the user as a label in the *buyATicket* form and as an input text in the *sellATicket* form. As already mentioned, in addition to the classification, we defined tagged values to be used as parameters in the application model. For example in Fig. 5, the *fName* that is associated with the

form (defined in the *Web Interface* (WI) domain, Fig. 1) indicates the form name, or the *key* that is associated with the table column (defined in the *Relational Database* (RD) domain, though is hidden in Fig. 3.) indicates the key type of this attribute in the database table. The motivation for that enhancement was that there is a many-to-many relationship between the elements in the various layers. In the case of the *ticket selling system*, the *buyATicket* form gathers elements from various classes: *Ticket*, *Event*, and *Order*, and each one of those classes elements can refers to several *Form*s or *Table*s.

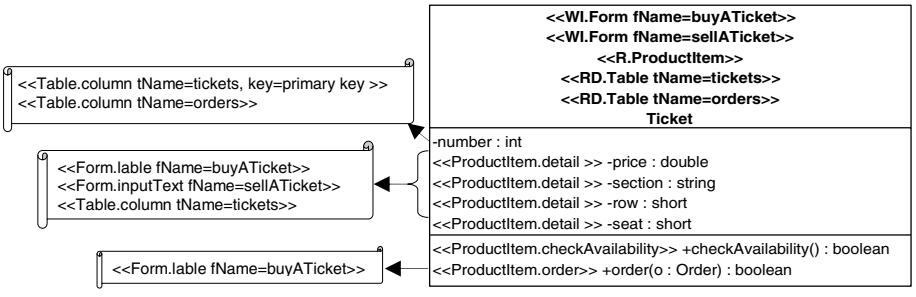


Fig. 5. The hidden classifications of the Ticket class are presented on the left

Having defined the application model, the developer should perform verification of the MCM-based application model with respect to the domain (reference) model. The verification of an application model is done separately for each domain. As described by the ADOM approach [15], the algorithm is performed in three steps: element reduction, element unification, and model matching. In the following we briefly present this algorithm.

In the ***element reduction*** step, classes that are not stereotyped by elements of the same domain model are neglected from each model separately. In the example of the *tickets selling application*, with respect to the *RD* domain the *User* and *Order* classes are omitted, with respect to the *R* domain the *Category* class is omitted, and with respect to the *WI* domain the *Buyer* and *Seller* classes are omitted.

During the ***element unification*** step, classes having the same domain stereotype are unified, leaving only one class in the resultant model. The multiplicity of that class denotes the number of distinct classes in the application model having the same stereotype. In the example of the *tickets selling application*, the resultant model of the *RD* domain consists of the *Table* class with multiplicity of 10. The resultant model of the *R* domain consists of 6 classes: *User*, *Provider*, *Client*, *Registration*, *Product*, and *ProductItem* all with multiplicity of 1. Finally, the resultant model of the *WI* domain consists of the *Form* class with multiplicity of 7.

In the ***model matching*** step, the resultant models of the previous step are matched against their corresponding domain models in order to verify the multiplicity of the elements. In addition the application model structure for each layer is verified with respect to the appropriate domain models. In the example of the tickets selling application the model adheres with the domain models which represent the layers of the system (i.e., the RD, R, and WI domains).

5 Summary

In this paper we have presented the Multi-Classified Model (MCM) approach for specifying layered-architecture based systems. The MCM approach extends a domain engineering approach called Application-based Domain Modeling (ADOM). Utilizing ADOM, MCM refers to each layer as a separate domain model, whose elements are used to classify the application model elements. Consequently, the application model is represented in a single model, which incorporates information from all of the layers. In this paper we have elaborated the first step of the approach, which deals with modeling a layered application from the structural point of view. Furthermore, we have briefly reviewed the second step, which deals with verifying the application against the predefined layers. We have illustrated the approach through a *ticket selling system* designed as a standard three layered architecture application.

We believe that the proposed Multi-Classified Model approach enhances the following:

1. **Evolution and Maintainability:** Since MCM represents the application model in a unified form, which incorporates information from all of the layers it can be easily modified as new requirements will not cause cascades of modification. In that case, the designer needs to manage a unified application model without losing the architectural information. Moreover, as it is easy to change the application model, the approach will prevent 'software decay' through time.
2. **Comprehension:** Software architectures enhance the ability to comprehend large systems by abstraction and separation of concerns. However, traditional modeling techniques, such as UML, do not support them very well. For example, in order to present the three layer architecture using UML the designer will have to describe three different packages and the relationships among them. The Multi-Classified Model approach simplifies the application model of a system with the layered architecture due to the use of a unified model for the application specification. Thus, the model is much more compact. In addition, as application model elements, such as classes, are classified according to their requirements to the appropriate layers (domains), we can easily understand the classification of each element with respect to the layers it belongs to. Thus, it seems plausible that the general understanding and readability of the application model will increase with respect to the traditional modeling techniques representation.
3. **Construction:** The defined domain models provide a partial blueprint for the development of the application model. The Multi-Classified Model approach typically documents abstraction boundaries between parts of an application, clearly defining which element belong to which layer, and constraining what parts of a system may rely on services provided by other parts. That helps to construct a good structure which is based on proven domain models.
4. **Verification:** Since the Multi-Classified Model approach uses ADOM, it provides verification to the application model according to the constraints imposed by the domain models which represent the layers. This cause the models to be valid with respect to best practices specified within the domain layer of ADOM. Furthermore, the Multi-Classified Model approach enforces the developers to preserve all the rules that were defined in the domain layers during the development.

It is worthwhile to mention that we have constructed a *student-courses registration system* under the same modeling infrastructure, showing that the approach can be used in different settings as well.

In the future we plan to formulate the Multi-Classified Model approach to support automatic generation of application code with the use of transformation rules that will be defined for each domain. This will support the third step of the proposed approach. In addition, we plan to extend the Multi-Classified Model approach to deal with dynamic aspects of the architecture as well. Furthermore, we intend to perform empirical evaluations in order to examine the benefits of the MCM approach, namely, model changeability, model comprehension, and model construction.

References

[1] Allen, R.J., Douence, R., Garlan, D.: Specifying Dynamism in Software Architectures. In: Proceedings of the Workshop Foundations of Component-Based Systems, pp. 11–22 (1997)

[2] Brooks, R.A.: A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation 2(1), 14–23 (1986)

[3] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture: A System of Patterns. John Wiley & Sons, Inc., Chichester (1996)

[4] Dijkstra, E.W.: The structure of the "THE"-multiprogramming system. Communications of the ACM 11(5), 341–346 (1968)

[5] Evans, E.: Domain-Driven Design: Tacking Complexity in the Heart of Software. Addison-Wesley Longman Publishing Co., Inc., Amsterdam (2003)

[6] Garlan, D., Monroe, R.T., Wile, D.: Acme: Architectural Description of Component-Based Systems. In: Leavens, G.T., Sitaraman, M. (eds.) Foundations of Component-Based Systems, pp. 47–67. Cambridge University Press, New York (2000)

[7] Luckham, D.C., Vera, J.: An Event-Based Architecture Definition Language. IEEE Transactions on Software Engineering 21(9), 717–734 (1995)

[8] Magee, J., Dulay, N., Eisenbach, S., Kramer, J.: Specifying Distributed Software Architectures. In: Botella, P., Schäfer, W. (eds.) ESEC 1995. LNCS, vol. 989, pp. 137–153. Springer, Heidelberg (1995)

[9] Malavolta, I., Muccini, H., Pelliccione, P., Tamburri, D.A.: Providing Architectural Languages and Tools Interoperability through Model Transformation Technologies. IEEE Transactions on Software Engineering 36(1), 119–140 (2010)

[10] Medvidovic, N., Rosenblum, D.S., Redmiles, D.F., Robbins, J.E.: Modeling software architectures in the Unified Modeling Language. ACM Transactions on Software Engineering and Methodology 11(1), 2–57 (2002)

[11] Medvidovic, N., Rosenblum, D.S., Taylor, R.N.: A Language and Environment for Architecture-Based Software Development and Evolution. In: Proceedings of the 21st International Conference on Software Engineering.ICSE 1999, pp. 44–53. ACM, New York (1999)

[12] Pastor, O., Molina, J.C.: Model-Driven Architecture in Practice: a Software Production Environment Based on Conceptual Modeling. Springer-Verlag New York, Inc. (2007)

[13] Pérez-Martínez, J.E.: Heavyweight Extensions to the UML Metamodel to Describe the C3 Architectural Style. ACM SIGSOFT Software Engineering Notes 28(3), 5 (2003)

[14] Reinhartz-Berger, I., Sturm, A.: Enhancing UML Models: A Domain Analysis Approach. Journal of Database Management 19(1), 74–94 (2007)

[15] Reinhartz-Berger, I., Sturm, A.: Utilizing Domain Models for Application Design and Validation. Information and Software Technology 51(8), 1275–1289 (2009)
[16] Roh, S., Kim, K., Jeon, T.: Architecture Modeling Language based on UML2.0. In: Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC), pp. 663–669. IEEE Computer Society, Los Alamitos (2004)
[17] Szyperski, C.: Component Software: Beyond Object-Oriented Programming., 2nd edn. Addison-Wesley Longman Publishing Co., Inc., Amsterdam (2002)
[18] Tanenbaum, A.S.: Computer Networks. Prentice Hall PTR, Englewood Cliffs (1985)
[19] Weigert, T., Garlan, D., Knapman, j., Møller-Pedersen, B., Selic, B.: Modeling of Architectures with UML (Panel). In: Evans, A., Kent, S., Selic, B. (eds.) UML 2000. LNCS, vol. 1939, pp. 556–569. Springer, Heidelberg (2000)
[20] Zarras, A., Issarny, V., Kloukinas, C., Kguyen, V.K.: Towards a Base UML Profile for Architecture Description. In: Proceedings of ICSE 2001 Workshop on Describing Software Architecture with UML, pp. 22–26. IEEE Computer Society, Los Alamitos (2001)

A Model Based Framework Supporting ITIL Service IT Management

Manel Jelliti[1,2], Michelle Sibilla[1], Yassine Jamoussi[2], and Henda Ben Ghezala[2]

[1] Service IntEgration and netwoRk Administration Team IRIT labs, 118, Route de Narbonne, 31062 Toulouse cedex 9, France
[2] RIADI-GDL labs, ENSI, University of Manouba 2035, Manouba, Tunisia
{jelliti,sibilla}@irit.fr
{jamoussi,Henda.BG}@riadi.rnu.tn

Abstract. The implementation of an adequate business management system for information technologies (IT) requires recognition of business needs, current level of maintenance, better insights into available approaches and tools, as well as their interoperability and integration. The approach we are proposing in this topic aims the reusing and the extension of CIM (Common Informational Model), a standard Model in System Management domain, for designing the ITIL (Information Technology Infrastructure Library) processes. According to ITIL, the CMDB (Configuration Management Database) forms the basis for effective and efficient IT Service Management. We will present how core processes correlate to each other and point out the challenge of setting up a CMDB. We also present the key requirements for designing CMDB with using the MDA (Model Driven Architecture) approach and focus in the PIM (Platform Independent Model) phase. In order to ensure a well-founded business management of ITIL CMDB and the different dependencies between processes, a static view must be provided with a behavior view. Our approach of the behavior modeling is based upon the integration of statechart diagrams UML2.0 in the CIM model.

Keywords: ITIL, CMDB, IT Service Management (ITSM), CIM, Model Driven Architecture (MDA).

1 Introduction

We attend for a few years to a revolution in the ITSM (IT Service Management) area. Indeed, the IT service management is not considering like purely technological but it also takes into account the business organizational aspect. This shift in the ITSM aligns with the principles of Business Service Management (BSM). In fact, BSM is now recognized as one of the most important attributes of a comprehensive systems management solution. Delivering relevant information to business decision makers is now a priority for management tool and application vendors. Recent trends in enterprise software have provided enterprises an opportunity to realize greater efficiencies and cost savings from their software investments. Additionally, these same opportunities have created new challenges for systems management software developers to keep

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 208–219, 2010.

pace with this dynamic environment. In this context, the IT Infrastructure Library (ITIL) [1] emerged like the de-facto-standard for IT service management. Indeed, ITIL gained his biggest popularity because it combines the principles of service- and process-orientation in IT Management and is easily accessible, it became increasingly attractive for IT organizations of almost any size, branch or organizational setup. The scope of ITIL is not limited to technical issues, but also covers the human and economic dimensions (business alignment) of IT Service Management. ITIL can be defined as guidelines of "Best Practices" that supports planning, monitoring and controlling of IT services. Constituted by a series of books, ITIL defines the whole of processes necessary for provision of IT services and provides rules of good practices. ITIL has vocation to establish a common vocabulary to the whole of IT industry actors and to propose a standard step of implementation of the IT services of organizations.

In this work we give a survey on the ITIL framework structure and its most important concepts and contents, including an outline of seven of ITIL's core reference processes. Furthermore, the paper discusses some important research topics related to ITIL, in particular Management Information Modeling and how reusing the concept of Network and System management for a Business management. The processes selected for this paper are Incident Management, Problem Management, Change Management, Release management, Configuration Management, Service Level Management, Availability Management and Capacity Management [1, 2]. Learn how these processes are related and how they can be integrated to a business system management. We show how the core processes correlate to each other and point out the central role of Configuration Management and the challenge of setting up a Configuration Management Database (CMDB) [2, 3]. Find out what makes a CMDB setup so difficult, which requirements a CMDB should fulfill and why the current commercial and scientific efforts address these challenges insufficiently. We also propose using the MDA (Model Driven Architecture) concept for building a business system management. Adequate tools are vital for a successful deployment of ITIL. But since ITIL is tool-independent and hardly formalized, sufficient and integrated tool support for ITIL is not available today.

2 Integration Process Scheduling

The service management in ITIL is described by ten processes and the Service Desk functionality, these last are grouped into Service Support Set (provider internal processes) and Service Delivery Set (processes at the customer–provider interface). Each process describes activity, functions, roles and responsibilities, as well as necessary databases and interfaces. In general, ITIL describes contents, processes at a high abstraction level and contains no information about management architectures and tools.

Likewise to other 'best practices' and standards, ITIL indicated some weaknesses [4]:

- Lack of holistic visibility and traceability from the theory (specifications, glossary, guidelines, manuals, etc.) to its implementations and software applications.
- Frameworks, best practices and standards are focused on logical level of processes, which instruct what should be done, but not how.
- Poor definition of information models corresponding to process descriptions.

Proposing a schedule to explain how ITIL processes should be integrated would be worthy for achieving the BSM level. The heart of BSM is the Configuration management process. It has the role to place at the disposal of all the other operational and strategic processes a database, the CMDB (Configuration Management DataBase). As mandated by ITIL, The CMDB is the repository for Configuration Items (CI's) in a given enterprise and the relationship between them. In most cases, whenever an ITIL service management process needs to access information outside its immediate scope of responsibility, this is supposed to happen through querying the CMDB. This information can refer to things quite different from IT infrastructure elements or services, e.g. artifacts of other ITIL processes like incident tickets, known error, RFC (Request For Change), but also records on information like customer and user data, whose control is usually not within the scope of IT management.

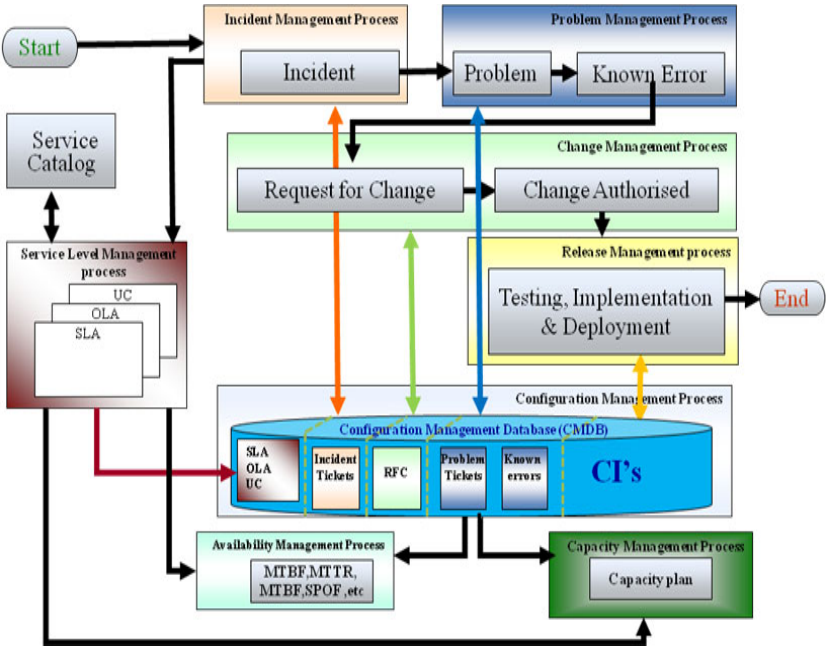


Fig. 1. The relationship between the different processes

We propose three phases to schedule the integration of the different ITIL processes:

- The first phase consists in the implementation of the Configuration management process. Indeed, all processes use the CMDB. In this phase the CMDB will be the view of the logical model of the IT infrastructure and IT services whose creation and maintenance is the main deliverable of the Configuration Management process. We could using the tools of network and systems management and integrate their data stores with the CMDB.

- In the second phase we should implement or/and integrate Incident, Problem, Change and Release management process. Indeed, these three processes will ensure the stability of the SI (Incident and Problem process) and the reliability of the CMDB (change and release management). For that we should include to CMDB all concerned process artifacts (Incident Tickets, known error, RFC, etc). If process supporting tools already exist we should integrate to CMDB the concerned external databases.
- In the third phase we should integrate Service Level Management, Availability and Capacity management process. For that we should include to CMDB the process artifacts of Service Level Management process (SLA, OLA, UC). Thanks to CMDB information, the availability and capacity management process provide availability indicators and capacity plan according to the Service Level Management process.

Figure 2 illustrates the different ITIL processes interactions, in resolving an incident that could occur in a real IT setting. We will use this scenario in the forth section, as support for presenting the concepts and building blocks described in this paper.

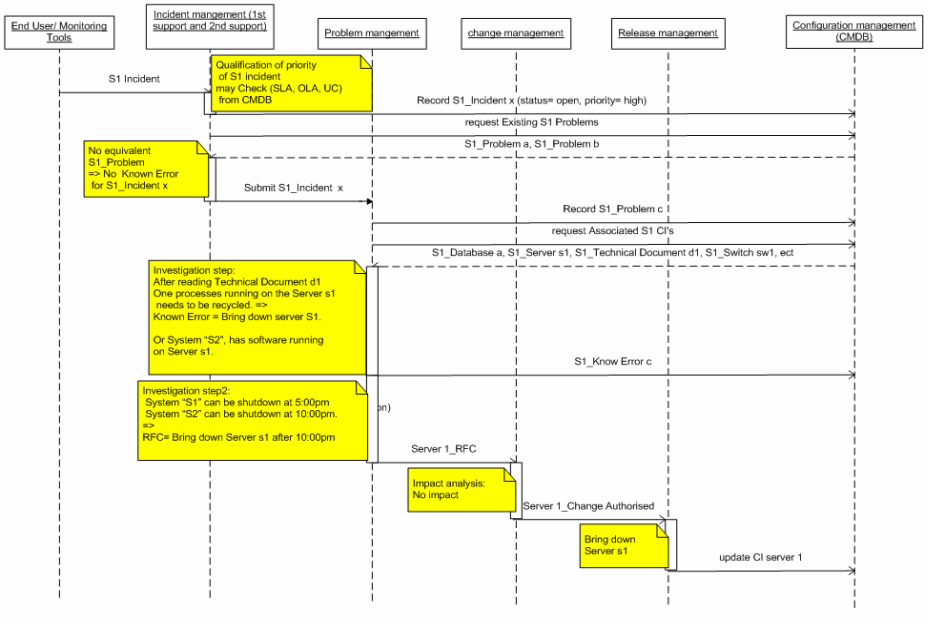


Fig. 2. Resolving incident steps

3 State of the Art

The advantages of the CMDB encourage the major editors of tools of administration to adopt this model. BMC, CA, HP Openview or IBM Tivoli. Each one tries at its rhythm to answer this waiting by gathering a scattered technological offer. Much

more than one simple database, a CMDB can be considered in the form of a distributed architecture, with a central base surrounded of software (supervision consol, service desk, inventory, etc) having their own repository. The editors seek to constitute only one collaborative repository, completely shared by the whole of the modules of their management tools. That supposes to reform the data model of the whole of the products. And in particular that of the supervision consoles, which have their owns models and repository. In the ideal, each tool registers directly the information that it is necessary for him to divide in the CMDB, the data strictly related to the activity of the tool (supervision, service desk, etc) remaining in the local repository. The multitude sources of information pose the problem of interoperability. That's why, editors have incorporated in their offer of CMDB advanced engines of reconciliation, which check the relevance of the data according to rules and thus guarantee a single point of indexing. Moreover, the interfaces of exchange and synchronization of data made their appearance. Thus, HP Openview rests on Connect-It, a tool inherited by the purchase of Peregrine. Connect-It disregards data model of the tiers repository by applying a pre-treatment of the data with transformation and rules of reconciliation. CA rests on its ETL tool for extraction of data (Advantage Data Transformer) to extract and reconcile the data of various sources. BMC uses the motor of reconciliation ARS (Action System Request) of its Remedy range and the middleware Enterprise Integration Engine which authorizes an exchange of the data in a bidirectional way with any application. Moreover, BMC envisages to develop a forty of native interfaces between their CMDB (BMCAtrium) [13] of second generation and the current software of the market: J2EE applications server, SAP software package, etc. All would go for best if the IT departments were always equipped in the same editor. However, it is seldom the case. Within large organizations, the difficulty lies in the meeting of existing bases of inventory owners in a single model of the data of configuration. Moreover, the editors protest all their will to adapt to existing.

4 Using MDA Approach for Designing CMDB

Generally, the greater parts of discussed criteria for CMDB tools have focused on functional requirements (e.g. visualization) and integration with other databases [3]. However, limiting assessments only to these requirements bears the danger of not addressing key standardization issues for CMDBs. Using MDA approaches could be a key standardization issues for CMDB built. The first step in the MDA approach [5] is the design of a model independent of any implementation platform (PIM). These models are specified in a formal standard to the OMG UML. In this step, the designer applications experts are released of implementation details and focus on the software specification itself. The second step consists in production of a specific model to a particular platform (PSM). This leads to consider technological specificities of chosen platform. In general, the PSM model is automatically generated from PIM model and then manually changed for optimization purposes. The last step is the generation of source code for a concrete implementation. The code obtained is obviously incomplete, but it will be a skeleton that takes into account the constraints modeled and will be a basis for final implementation. Using the MDA approach for designing CMDB implies the choice of a flexible data model to ensure the requirement of CMDB.

We note that a big part of Configuration Items (CI's) that should feed in the CMDB (e.g. infrastructure) already exists in the management tools databases. Therefore, reusing existing management models in designing CMDB PIM step seems to be a good approach.

4.1 A Flexible Data Model

There are many different types of Configuration Items (CIs), from computer systems to network hardware including software servers and process artifacts. Without a data model that accurately reflects these types and the relationships types that can exist between them, the CMDB could store attributes that do not pertain to their CIs, leave out necessary attributes, and make it harder to search for groups of CIs. This data model must be both object-oriented and extensible. In fact, the benefits of an object-oriented data model include enforcement of common attributes among similar types of CIs and the ability to search within not just a given class of CIs, but within any branch of the hierarchy. If the data model has one base class from which all others are sub-classed, we can search for all CIs and their relationships. The necessity of an extensible data model is due to nature of the infrastructure and its underlying technology in constantly changing. That means the types of CIs and relationships in a CMDB must also change, so we need a data model that is extensible. The Common Informational Model CIM [7] model proposed by the Distributed Management Task Force (DMTF) assures the needs previously mentioned. Indeed, with the set of classes suggested in CIM model, we are able to represent every CI (configuration item) which will populate the CMDB and the relationship between them (e.g. CIM_Component, CIM_Dependency). Moreover, in agreement with MDA, part of CIM is textual, human-readable language (Managed Object Format (MOF)) for describing modeling constructs that can be processed by automated tools. In 2006, Brenner and al. established a set of criteria for evaluate the possible reuse of CIM to build a CMDB [6]. They give a comprehensive overview of the criteria and illustrate how CIM fulfill these criteria. Despite the flexibility of CIM, they conclude that this last present a lack of class for the representation of process artifacts and a lack of means for the integration of external databases. However, we do not agree with Brenner's CIM analysis. In fact, we have identified a set of CIM classes that allow the representation of process artifacts. Moreover, the DMTF has initiated a CMDB federation working group which aims to propose means for the integration of external databases.

4.2 The Representation of Process Artifacts

CIM aims to providing a common way to represent information about networks and systems as well as services. It defines managed resources as object classes that can be further refined by means of strict inheritance. Since all CIM classes derive from the managed element class as defined in the Core Model [8], CIM provides a coherent view on the modeled infrastructure. On other hand, for building CMDB we should include the linkage of ITSM processes to infrastructure elements. In particular, key concepts of ITIL such as Incident records. That is why; we have tried to find which

CIM classes could respond to this need. In this context, we have identified the CIM Support schema [9, 10]. These models are referred to as the Problem Resolution Standard or PRS. The primary purpose of PRS is to define an open exchange standard that facilitates Solution exchange and Service Incident processing between cooperating parties, both within an organization and across organizational boundaries. All classes defined by the specification begin with the letters "PRS_" as opposed to the habitual use of "CIM_" for other CIM-related object models. This prefix is due to historical reasons and all classes within PRS are to be treated as standard CIM extensions. We illustrate in table 1, the different CIM classes that could be used to represent CI's.

Table 1. The CIs/ CIM classes' mapping [9, 10, 11]

CIs/Process artifacts	Root node CIM Class	Schema
Incident Record	PRS_ServiceIncident	CIM Support
Problem Record	PRS_Problem	CIM Support
KnownError	PRS_Resolution	CIM Support
RFC	PRS_Transaction	CIM Support
Change Accepted	PRS_Solution	CIM Support
SLA	PRS_Agreement	CIM Support
Person	PRS_Contact	CIM Support
Service	CIM_Service	CIM Core
Application	CIM_ApplicationSystem	CIM Core
Software	CIM_SoftwareElement	CIM Core
Hardware	CIM_ComputerSystem	CIM Core

4.3 Toward Dynamic Process Artifacts Management

Even if, the CIM Support schema proposes a set of classes to express process artifacts there is no means to automatically ensure their life cycle in ITIL orchestration processes. Indeed, an incident will be qualified by many states throughout its lifecycle (e.g. Open, In-resolution, Close, etc). It is the same needs for other process artifacts. CIM lacked a model of behavior to automate the life cycle of different CI. One technique used to add behavior to model CIM is the integration of state machines of UML. This technique is consisting to associate a state machine with a CIM class [12]. Each state in this diagram represents the possible states in which the class may be, and each transition indicates changes of states. Each transition is composed by an event, a condition and a possible action to achieve. The event is the trigger for the verification of a condition that will induce a probably action. The CIM classes using to ensure this mechanism are *CIM_IndicationSubscription*, *CIM_IndicationFilter* and *CIM_ListenerDestination*. To present our approach we will use the scenario presented in section 2. We design in a first step, the static view of CMDB (Figure 3). However,

based on this scenario we detect a lack in CIM Support schema, there are no associations between classes that represent the process artifacts (e.g. incident record, problem record) and the class *CIM_ManagesSystemElement* that represents the CI's of infrastructure. That's why, we added in the static view, two new associations (*Associated Incident* and *Associated Problem*). There is also a lack of association between the owner of a process artifact *PRS_Contact* and his role in the organization, so we also added a third association (*Associated Organization)* to ensure this ITIL requirement. Finally, we extend CIM with a new property (*BusinessStatus)* in order to trace all states that a CI could have in his life cycle. Indeed, we don't find in CIM a property that expresses a business status of a CI, we just find property like OperationalStatus which correspond to an operational state of a *ManagedSystemElement*.

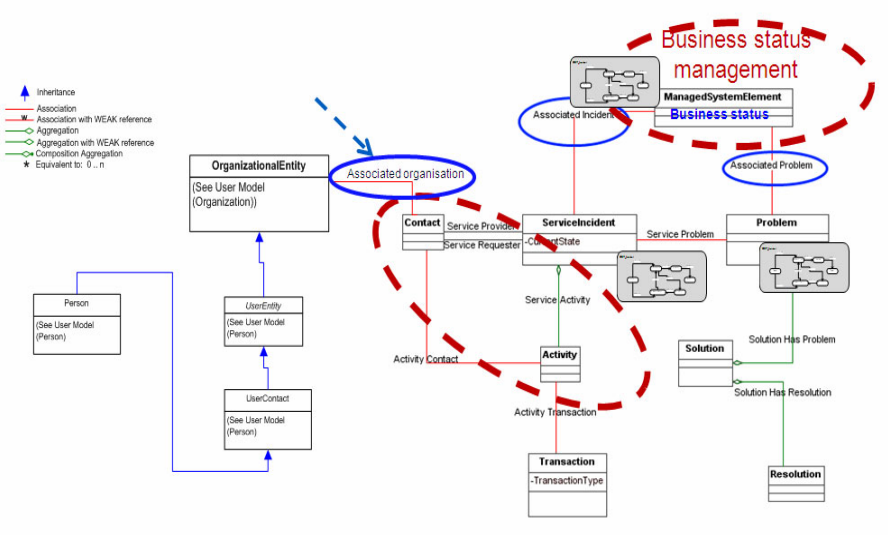


Fig. 3. The static view: The Informational and Organizational model proposed (M1 level)

The second step is the designing of CMDB's dynamic view, for that we associate behavior to each adequate process artifact's class and use subscription mechanism to adequate events to automate and ensure dynamicity management. The formalisms using to express behavior is the UML state machines. In Figure 4 and Figure 5, we illustrate the using of behavior to orchestrate a part of the scenario workflow. We just focus in the management of the current state of a ticket incident.

Figure 5 illustrates a simplified State machine (DET_Incident) that we could associate to *CIM_ServiceIncident* Class to maintain the Current State of the incident tickets (record). In this DET , we consider that there is no condition to verify and an action A_i is the updating of the current State (e.g. *PRS_ServiceIncident.CurrentState := Problem_Submitted*). In the example of events CQL (Code Query Language) specification described below, an instance of *CIM_ServiceIncident* is replaced by *$currentInstance$.*

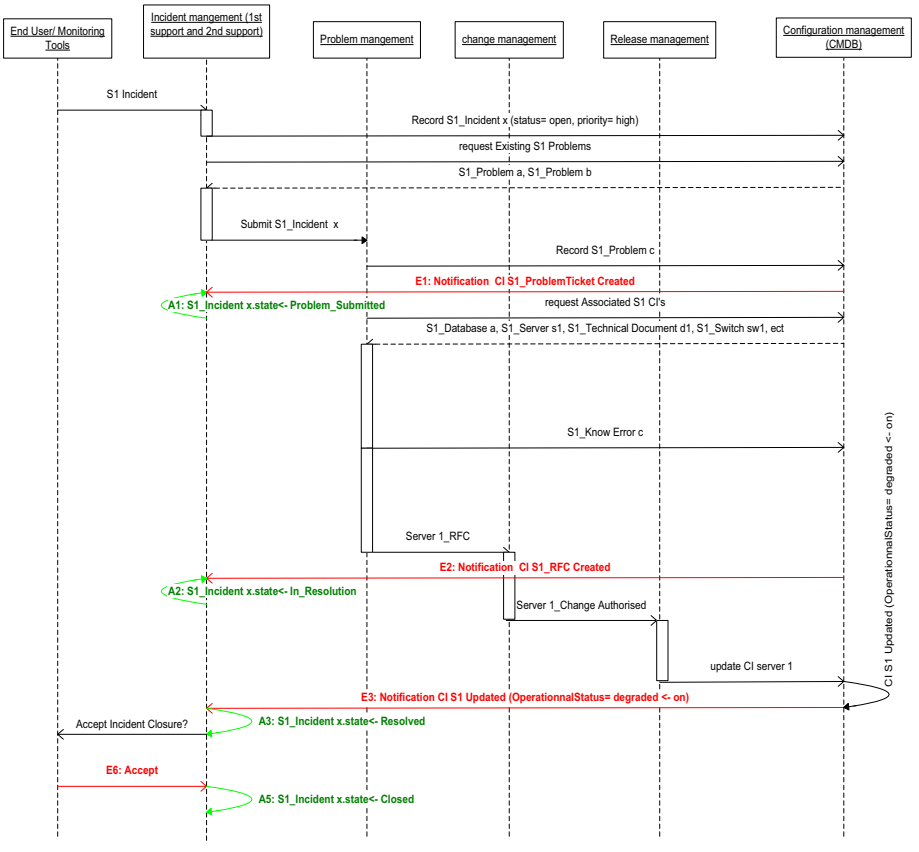


Fig. 4. Exemple of Incident Current State update

E1 CQL Specification

```
SELECT * FROM CIM_InstCreation, AssociatedProblem AP
WHERE
AND AP.Antecedent = CIM_InstCreation
AND AP.Dependent = $currentInstance$
```

E3 CQL Specification

```
SELECT*FROM CIM_InstModification, AssociatedIncident AI
WHERE
AND
AI.Antecedent = CIM_InstanceModification.SourceInstance
AND AI.Dependent = $currentInstance$
AND
CIM_InstanceModification.SourceInstance.OperationnalSta
tus = On
```

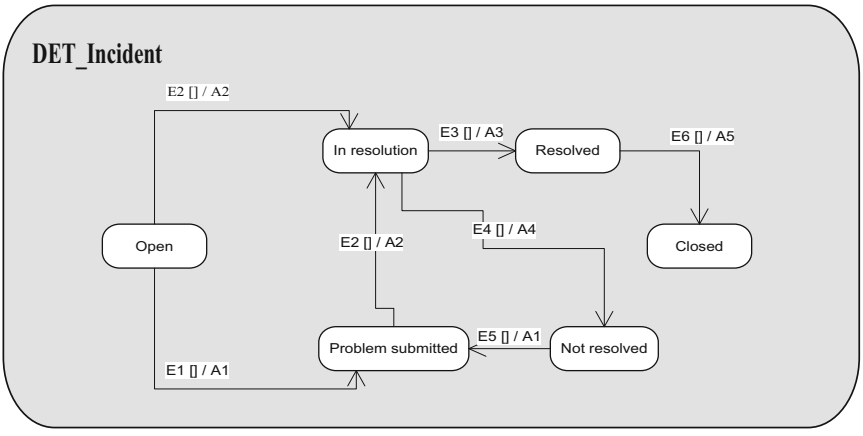


Fig. 5. Dynamic Management of CIM_ServiceIncident

5 Integration with External Databases

In this section we will present the architectural view of our proposed framework and introduce a part of the PSM step by considering the technological specificities of the chosen platform.

A CMDB should be part of a configuration management system. It may integrate other management data repositories (MDRs), including other CMDBs. The usefulness of a CMDB is dependent on the quality, reliability and security of the data organized through the CMDB. In practice this goal is challenging because the management data are stored in MDRs that use different data models and that support different access interfaces. Examples of potential MDRs include CIMOMs, vendor tools, and customer in-house data stores.

A solution that assumes the conversion of all data to a single data model or consolidation of all data in a single repository is neither practical nor desirable. What is needed is a solution to federate heterogeneous MDRs, including linking together all the data about an IT resource, even when the data for a resource may be dispersed across multiple MDRs. IT resources include configuration items (e.g., computers, software, services, buildings), process artifacts (e.g., incident records and request for change forms), and relationships between them. Each resource, including relationships, may have a separately managed lifecycle and its state may be represented by a set of properties. The DMTF have established a Working Group to treat about in problems of CMDB federation. The goals of the CMDB Federation Working Group are to define a platform independent, industry standard specification that:

- ✓ Defines the XML schema for federating management data repositories in a model and protocol neutral fashion.
- ✓ Defines encapsulating XML elements for items (Configuration Items and/or process artifacts), relationships, and data records associated to the items and relationships. The data model of the encapsulated data is not defined.

- ✓ Defines a query interface and expression format through which clients may request item and relationship data from CMDBs and other management data repositories, and that facilitates queries involving the navigation of graphs consisting of items and relationships.
- ✓ Defines protocol-specific bindings, reusing existing standards where applicable.
- ✓ Defines methodologies and interfaces for accessing data sources and their capabilities.

The working group will also provide non-normative information that describes ways the specification may be implemented and used, and best practices for federating disparate data models. In October 2007, a first Draft of CMDB Federation specification was published to provide a solution to the federation and integration aspects.

This DMTF initiative encourage us to respect the WBEM (Web-Based Enterprise Management) architecture for building the architectural view of the CMDB (Figure 6) and we plans to start the implementation of our CMDB in OpenPegasus one of the open-source implementation of the DMTF CIM and WBEM standards.

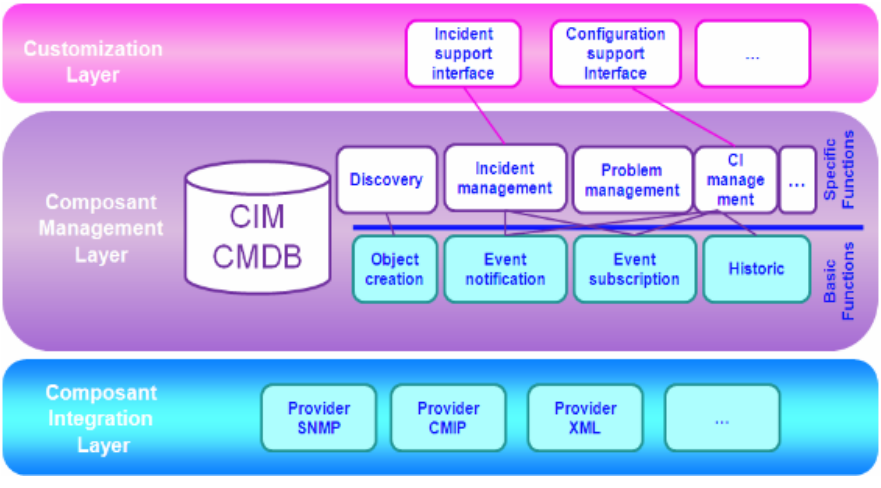


Fig. 6. The proposed Architecture

6 Conclusion and Perspectives

This paper presented a set of concepts for integration of ITIL processes. In a first step, we propose a plan to integrate ITIL processes. We consider the enterprise IT management maturity as criteria to scheduling this plan (from operational management level to service management level). The focal point for this integration is the CMDB and which elements will be integrated to this latter. In a second step, we propose to reuse and extend the standard management model CIM for designing CMDB. Indeed, we establish the static view of CMDB thanks to the identification of appropriate CIM classes which represent CI's, process artifacts and the different relationships between

these elements. Afterwards, we enriched this static view by a dynamic view which ensures the corresponding behavior of ITIL philosophy. To conclude, we point out that the work proposed is a specification of CMDB and ITIL concepts. Our perspective would be to move towards an implementation of this CMDB. For that, we should have an execution platform with basic services such as subscription, notification event and automatic updating of CI's. Using MDA concepts allows us to rely on tools transformation models to generate specific model to a target platform.

References

1. Office of Government Commerce OGC (ed.).: Introduction to ITIL. IT Infrastructure Library. The Stationary Office (2005)
2. Office of Government Commerce OGC (ed.).: Service Support. IT Infrastructure Library (ITIL). The Stationary Office, Norwich, UK (2000)
3. Colville, R.J.: CMDB or Configuration Database – Know the Difference. Gartner (2006) Research Note G00137125
4. Strahonja, V.: Definition Metamodel of ITIL. In: Information Systems Development (ISD 2009): Challenges in Practice, Theory. Springer Science Business Media (2009)
5. OMG: Model driven architecture (MDA). Document number ormsc/2001-07-01. Architecture Board ORMSC1 (2001)
6. Brenner, M., Garschhammer, M., Sailer, M., Schaaf, T.: CMDB - Yet Another MIB? On Reusing Management Model Concepts in ITIL Configuration Management. In: International Workshop on Distributed Systems: Operations & Management (DSOM 2006). IFIP/IEEE (2006)
7. DMTF: Common Information Model (CIM) concepts Version 2.4 (2003)
8. DMTF: Core MOF Specification 2.21 (2009)
9. DMTF: Solution Exchange and Service Incident Specification White paper 2.5 (2002)
10. DMTF: Solution Exchange and Service Incident Specification MOF Specification 2.21 (2009)
11. Jelliti, M., Sibilla, M., Jamoussi, Y.: Model Extension and Automation Requirements in Configuration Management Process. In: International DMTF Academic Alliance Workshop on Systems and Virtualization Management (SVM 2007): Standards and New Technologies, Toulouse (2007)
12. Jocteur-Monrozier, F., Bianchi, S.C.S., Sibilla, M., Vidal, P.: « Contrôle Dynamique des Systèmes Complexes par la Modélisation Comportementale ». J. Revue Génie Logiciel, Hermès 72, 14–21 (2005)
13. BMC Software: BMC Atrium CMDB, Enable the core of your IT infrastructure. Product DataSheet, 61966 (2006)

A Structured Evaluation to Assess the Reusability of Models of User Profiles

Lillian Hella and John Krogstie

Dept. of Computer and Information Science
Norwegian University of Science and Technology, 7491 Trondheim, Norway
{hella,krogstie}@idi.ntnu.no

Abstract. In the creation of an adaptive mobile personalisation system it is useful to investigate whether existing models are applicable. Such models are usually structured as ontologies. We view existing ontologies from a reuse perspective, and have chosen to specialise the SEQUAL quality framework for evaluation of existing models. SEQUAL has previously been used for the evaluation of modelling languages and approaches, including the evaluation of ontologies. Using the semiotic quality categories in SEQUAL, an evaluation has been made of potential ontologies. The result of the evaluation is that none of the evaluated ontologies satisfies requirements as models that can be reused or built on, and the profile ontology has been created from scratch.

Keywords: Profile, ontologies, reusability, evaluation, quality of models.

1 Introduction

The vision for the next generation Web as the semantic Web [1], is now often combined with Web 2.0 technology to predict Web 3.0. Information is accompanied by metadata about its interpretation, so that more intelligent and more accessible information-based services can be provided. A central component in the semantic Web and its applications is information modelled as ontologies. An ontology can be seen as an explicit representation of a shared conceptualisation [2] that is formal [3], and will encode the semantic knowledge and enable sophisticated services.

The first step of creating an ontology is to define the domain. The next step is to consider reuse of existing ontologies [4], to see if they can be reused as is or as a basis for customisation. Our goal for this work has been to investigate the possibility of reuse or building on ontologies in the domain of personalisation of mobile services. Similar ontologies have been evaluated in [5]. However, this evaluation is not using a structured evaluation framework. A classification of evaluation methods for ontology quality is presented in [6]. Five aspects for ontology quality are mentioned: syntax, vocabulary, structure, population of classes and usage statistics. Existing approaches cover at most three of these. We do not find these five aspects sufficient for the evaluation of reuse of ontologies with regards to the domain an ontology is modelling. Our approach is to specialise the model quality part of the semiotic quality framework SEQUAL [7]. SEQUAL has been used for similar evaluations in a number of related

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 220–233, 2010.

areas, such as ontologies [8], ontology languages and tools [9], object-oriented modelling languages [10], goal-models [7], requirements models [11], ontology building methods [12],[13] and interactive models [14]. As in our work, SEQUAL has in these cases been used by specialising the generic framework to the relevant domain and goal of modelling.

The rest of the paper is organised as follows. First, personalisation and needs for the profile ontology are described. Then, the specialisation of SEQUAL is presented. Third, the existing ontologies to be evaluated are described. Applying SEQUAL, the quality evaluation of the ontologies is presented. Finally, conclusions are drawn.

2 Personalisation and Profile Ontology Representation Needs

The main goal with personalisation is to improve the user's experience of a service. Personalisation is often needed to overcome information overflow and is important for service providers to acquire better knowledge of their end-users and for achieving improved business results [15]. Personalisation usually requires the user to directly interact with a service, and any preferences are usually kept by the service provider and not the user. However, it is essential for the decision on personalisation to refocus from the service provider to the user if the personal preferences of the user depend on context and one wants to protect the privacy of the user [16]. One of the main challenges for future personalised support lies in the combination of public and private information, and the combination of personalisation and contextualisation [17]. To provide personalised mobile services, different types of information are useful. Here we focus on users' personal profiles. The profile contains all the information related to a person as an actor, its goals etc., and follows the user everywhere independently of the context. Then we have information about the capabilities of the mobile device (as described in e.g. W3C Delivery context ontology [18]). The environment of the person using the device will we term the context of use. We note that many that work on mobile applications include parts of the information we have in the personal and device profiles in the term context (e.g. in relation to the definition of context by Dey and Abowd [19]), but we find it fruitful to more clearly distinguish these terms, since the profile information follows the user as he change context.

In the work leading up to the need for ontology reuse, we have looked in particular on a case of personalised information support for food shopping. Even though we have focused on a specific domain, with personas and scenarios with characteristics related to this, the concept of a personal profile with regards to how it is to be communicated with the world is what we want to evaluate. We have categorised the different types of information we find necessary for such a profile to model. The information that is to be captured in the personal profile can be divided in three main parts. 1: Personal information consists of categories of information that is common for all users. This change very seldom and typical examples are name, birth date and address. 2: Stable interests. It is called stable because the type of information does not change frequently, due to importance and relevance. Once a user has an interest, he is likely to have this interest for a longer time span, e.g. favour a specific producer of jam, finding it positive that food is ecologically produced or price is not considered crucial. Because of the personal value of expressing this and keeping this type of information updated, a user

would typically have motivation to do this by himself if it would change. 3: Temporary interests. For a limited time period a user could be interested in for example buying a new digital compact camera. In our case the daily shopping list represents the temporary interests, so it should be possible to create a shopping list. As soon as the goal is fulfilled, it is no longer part of the personal profile.

3 Evaluation Framework – SEQUAL Specialisation

We view ontologies as models that can be reused or built on, and apply a framework for evaluating model quality, SEQUAL [7], to provide a systematic analysis of the quality of ontologies. In this section we will present the specialisation of SEQUAL.

The main concepts or sets of the SEQUAL framework and their relationships are described below. For each set we refer to the ontology to be evaluated in general and specify what the desired qualities are:

G – The goal is reuse as is or as a basis of an existing ontology in the relevant domain (see **D** below) to be able to provide personalised services.

K - The knowledge of the evaluators.

L - The language the profile ontology is represented in.

M - The model (ontology) to be evaluated.

D - The modelling domain covered by the evaluated ontology. We need an ontology that supports the description of a user to be able to receive personalised services.

T - All the statements in the ontology represented and interpreted by a tool.

I - Social actor interpretation is the set of all statements which the externalised model consists of, as perceived by the evaluator.

The quality categories described next are used as requirements for the evaluation:

- **Physical quality.** The ontology should be physically available and it should be possible to make changes to it. An available ontology should be possible to open in an ontology editor. In this way it will be possible to view and access the ontology, and further make changes to it if necessary. We have decided to use Protégé [20], a free, open source ontology editor with an active community.
- **Empirical quality.** If a visual representation of the ontology is provided it should be intuitively and easy to understand. It is an advantage that the structural quality of the ontology is good. High empirical quality will support the achievement of pragmatic quality.
- **Syntactic quality.** The ontology should be represented according to the syntax of a preferred machine readable language. More specifically, we prefer OWL DL. It is a W3C recommendation, and provides ensured decidability. WonderWeb OWL Ontology Validator [21] has been used for OWL sublanguage specification.
- **Semantic quality.** The ontology should cover the area of interest fully or partially, as specified in section 2, so that it is possible to easily extend it to do that (completeness). It could also be possible to take out a subset of the ontology so that it does not cover more than what is necessary (validity). Terms used in the ontology should be congruent with words used in the domain. It is important that there is good correspondence between the concepts needed and the ones provided by the existing ontology. Our goal is not to create a new standard personal profile, but to

be able to reuse an already tested and used model. Semantic quality is the most important quality category in this evaluation.

- **Pragmatic quality.** It should be possible to understand what the ontology contains, and being able to use it for our purpose. The pragmatic quality category also includes provision of necessary documentation. Documentation that is easy to understand is advantageous, and it should be consistent with the actual ontology.
- **Social quality.** The ontology should have a relatively large group of followers and parts should be used by other ontologies (this can be judged through metadata, e.g. recognition annotation, efficiency annotation).
- **Organisational quality.** The ontology should be freely available and accessible through a freely available tool. It should be available in a standard format and it should be available and supported for the coming years.

4 Existing Profile Models and Ontologies

An upper ontology [22] is an attempt to create an ontology describing general concepts that are equal across all domains. The aim is semantic interoperability between ontologies created under such an upper ontology. A domain ontology [22] models a specific domain, and represents the knowledge about the domain. We have considered both. We have only looked into ontologies that are publicly available and referenced in papers. Ontologies only mentioned in papers [23],[24],[25],[26],[27] are not considered. In the following sections we will describe the ontologies assessed.

4.1 FOAF

The Friend Of A Friend (FOAF) [28],[29] ontology has a simple vocabulary for describing people, what they do and their relations to other people. Hence, it is often used for describing people's social connections and networks. FOAF is represented using RDFS and OWL [29],[30]. FOAF is a popular, much used, and discussed use of semantic Web technology [31]. Its popularity is evident from related activities. A set of communities and projects are mentioned at their project Web page, but are not described in detail and not linked to. In addition, FOAF related news is available at delicious [32]. Other communication channels are IRC and mailing lists.

The terms defined are categorised as *FOAF basics, Personal info, Online accounts/IM, Projects and Groups* and *Documents* and *Images*. FOAF is situated around the class *Person* [33]. The FOAF vocabulary is intended to be uncomplicated, pragmatic and designed to allow simultaneous deployment and extension. FOAF is intended for wide scale use. Personal information is made accessible by having people publishing information about them in the FOAF format. When the person information is published, machines will be able to use it. FOAF core is considered stable [29].

4.2 OpenCyc

Cycorp [34] provides the Cyc technology, for intelligence and reasoning. Cycorp has an open source version of the knowledge base, called OpenCyc [35]. It consists of hundreds of thousands of terms, together with millions of assertions that relate terms

to each other. The OpenCyc upper ontology covers the domain of all of human consensus reality. Since it tries to cover everything in the world, the ontology is large.

The formal language CycL is used to represent the original version. Its syntax derives from first-order predicate calculus. The knowledge base consists of modules that are called microtheories. Such microtheories are a kind of subontologies. This knowledge base is possible to download, and accessed through a Web browser. However, OpenCyc is now also represented in an ontology language and OWL versions of the OpenCyc ontology can be downloaded.

Online concept browsers [36],[37] are available. In [36] concepts are separated into collections and predicates. A search result is a written definition, of the term, together with its unique tag and aliases. In addition super concepts, sub concepts, and instance of concepts are listed. The concept browsers have no tree structure to view the relation between concepts. There is also a more general OpenCyc blog [38].

4.3 SUMO

The Suggested Upper Merged Ontology (SUMO) [39] was created as part of the IEEE Standard Upper Ontology Working Group (SUO WG). SUMO consists of definitions that are intended for general purpose terms and wants to be a basis for domain ontologies that are more specific [40]. The original SUMO is specified using SUO-KIF [41], Standard Upper Ontology - Knowledge Interchange Format, and is a simplified form of the knowledge representation language KIF. Later SUMO has been translated into OWL. SUMO has been referenced in many papers independent of its funders. A selection of referenced papers are presented in [42].

SUMO's initial goal was to construct a single, consistent, and comprehensive ontology. Now SUMO has been put together with Mid-level Ontology (MILO) and several domain ontologies. Domain ontologies that are included are for example the ontology of Communications, Countries and Regions, Economy, Finance, Geography.

It is possible to browse the content of the knowledge base in their online browser. SUMO is connected to the WordNet lexicon [43],[44]. In the SUMO online browser [45] one can navigate from a SUMO concept to the corresponding WordNet term. In SUMO, a person is modelled as class *Human* which is equivalent to WordNet's person as a human being or a human body.

SUMO consists of around 1000 well-defined and well-documented concepts [46],[45]. The concepts are interconnected into a semantic network together with a number of axioms. The class hierarchy can be viewed in [47]. The axioms are common-sense notions that are generally recognised among the concepts. Open source toolset for browsing and inference can be downloaded with KIF knowledge engineering environment [48].

4.4 GUMO+UbisWorld

GUMO (General User Model Ontology) developed in OWL is made for the "uniform interpretation of distributed user models in intelligent Semantic Web enriched environments" [49]. GUMO is related to UserML (User Model Markup Language), which is a RDF-based exchange language for user modelling between decentralised systems [50]. The GUMO ontology can be integrated with ubiquitous applications with the UbisWorld user model service.

The main focus of the UbisWorld [51] approach lays on research issues of user modelling, ubiquitous computing and semantic Web. UbisWorld can also be used for simulation, inspection and control of the real world. UbisWorld is a version of GUMO that includes additions that can be used for the ubiquitous computing area.

GUMO+UbisWorld have the following basic user dimensions with differing time span: *emotional state, characteristics* and *personality*. A user model service manages the information about users and gives more advantages than a user model server would do. GUMO and UserML together focus on creating a common language and ontology for communication of user models [52].

gumo.org [49] and ubisworld.org [51] share a common interface. To get access to provided features it is necessary to sign up as a member. The ontologies in different versions, among them static and dynamic versions, exist. The Ontology Bowser presents a set of ontologies that can be viewed as foldable class trees. External ones (e.g. SUMO, GUMO, OpenCyc) can also be viewed.

5 Ontology Evaluation

In the next sections the evaluation is presented. It is a result of a comparison of what the ontologies are and represent, and the expectations for the different quality categories according to the desired personal profile ontology specified in section 3.

5.1 FOAF

Physical quality. FOAF is available with a vocabulary specification and an OWL-file. The OWL file opens in Protégé, and changes can be made.

Empirical quality. FOAF is not presented visually in the information found, and there is no overview figure provided. Even though FOAF is a small ontology with relatively few classes and relationships, it would be advantageous with a graphical representation of how the concepts relate. Classes and properties are described in writing and with some practical examples. Access to much related information is available from main Web page.

Syntactic quality. The OWL validator classifies FOAF to OWL Full.

Semantic quality. FOAF covers the class *Person* that is disjoint with *Project*, *Document* and *Organization*. The *Person* class has amongst others these properties: *family_name, firstName, surname, gender, geekCode, interest, knows, made, maker, publications, workInfoHomepage, birthday, dnaChecksum, name, phone, homepage, isPrimaryTopicOf, msnChatID, assurance*. Most of the properties are related to the online world of a user, and not the life as a physical actor. Some are also not relevant, e.g. the property *interest* that in FOAF implies a persons interest in a document. Parts of the basic types of information correspond to our definition of personal information. However, it is not complete. Little of FOAF is superfluous. For example, in relation to recommendation solutions, information or relations to other persons could be relevant. There are several aspects from the person domain that are missing.

Pragmatic quality. Documentation of FOAF terms is provided. Classes and properties are described in relation to what they are used for. There is documentation for the decisions that has been taken and explanations to the created ontology. Examples for each class are available, and make it easier to understand what a class or concept is intended for. The reference to the OWL file could be more visible. Some of the names of the properties are not very intuitive.

Social quality. FOAF have a relatively large group of followers. It is possible to chat about FOAF related subjects at an IRC channel #foaf. FOAF also has a mailing list and is used in many social networks and projects. FOAF is included as a part of GUMO.

The organisational quality. FOAF is freely available. It is available in a standard format, and will probably be available and supported for the coming years.

5.2 OpenCYC

Physical quality. OWL-files are freely available in a downloadable zip-file. The OpenCYC ontology is very big, and was too large to open in Protégé. Online concept browsers made it possible to search and view concepts.

Empirical quality. OpenCyc does not have any visual representation, hence it is difficult to get an understanding of the content of the ontology. Concepts are described in writing, but the relation to other concepts can not intuitively be discovered other than through direct relations to other concepts.

Syntactic quality. The OWL sublanguage used has not been detectable because of the file size.

Semantic quality. Difficult with a more detailed analysis because it could not be viewed in Protégé, the concept browser does not show the information hierarchically, and manual inspection is difficult. Therefore, this is based on the written documentation and the concept browser. The initial impression of the concept *person* is that there could be an overlap with our needs. However, as only class and axioms and not attributes are included in the concept browser, its completeness is hard to assess. Also, it is difficult to understand the model based on the OWL file itself.

In general, it seems like OpenCyc describes more about the world than we need, e.g. OWL constructs (e.g. owl class, owl datatype property) and CycL terms have been specified in the same way as other OpenCyc concepts. Hence, it seems like concepts are modelled at a very low level. In addition, it would be difficult to extract the parts that were needed because of all the dependencies and the size of OpenCyc.

Pragmatic quality. There is limited documentation for how to use and understand OpenCyc. Documentation about the original OpenCyc model is available, but it would take great effort to become familiar with it. It is not clear whether the information is intended for the original version, the OWL version, or both. Training material exists, but is for the use of the original version only. Parts of a handbook are not available (e.g. section 7), and the last update was done in 2002. The concept browsers do not give much insight into the model, other than very long written descriptions

which only give an overview. Missing hierarchical representation is a drawback, as one can not view the entire structure. The separation of concept in collections and predicates in [36] is not explained, and what kinds of concepts belong to which term is not clear. All these factors, and the lack of user manual intended for the OWL version, makes it difficult to start using OpenCyc.

Social quality. It is difficult to know how many followers there are and to what degree the OpenCyc ontology is used in other projects. They have a discussion forum. However, this is a forum with little activity. OpenCyc also have an IRC Channel and a blog with little activity. Last update was November 2008. However, it seems like the original OpenCyc has higher priority than the OWL version. All these social forums are mostly intended for the original version.

The organisational quality. The OpenCyc ontology is freely available in a standard format, and will probably be available and supported for the coming years. Might be stable, but updates are probable.

5.3 SUMO

Physical quality. SUMO has been translated into OWL, and a selection of domain ontologies is also included. Opens in Protégé.

Empirical quality. The visual overview does not separate SUMO, MILO, and domain ontologies. The complete class hierarchy makes it possible to view all the relations between classes. Because of its size, crossing lines in the figure are inevitable. The figure is still readable, and ok to navigate.

Syntactic quality. The OWL validator classifies SUMO as OWL Full.

Semantic quality. The figure depicted in the description about SUMO and the class hierarchy does not correspond with the classes found in the OWL file. SUMO is detailed enough to include the class *Human* that corresponds to our concept of a person. Other classes that could be relevant are also included. *Human* is a subclass of both *Hominid* and *CognitiveAgent*, and has subclasses *Man* and *Woman*. These classes have a number of properties. Large parts of the ontology are irrelevant (poor validity). The same applies to attributes. Neither personal information, stable interests, or temporary interests are fully supported. More constructs are necessary to be able to cover our domain fully. In general, the most visible overlap is in connection to the leaf nodes we have mentioned, and not so much to the higher level concepts.

Pragmatic quality. In the Protégé tree structure there are two other classes on the same level as the class Entity, which are left out of overview figures. We do not know for what reason. Papers referencing to SUMO give an easy to understand overview of the upper levels of SUMO. The specific domain ontologies that have been included are not described, but seem to be an integrated part of SUMO. No tutorial or user manual is found for the ontology. Material found relates only to the KIF version of SUMO and tools. Several of the attribute names are not intuitive, and do not indicate which classes or types the relation connects. Also they say little about the direction of the relationship (e.g. whether a man IS a son, or HAS a son).

Social quality. SUMO is assumed mature since there is little activity related to it. We have not found documentation covering how much used it is in other projects.

Organisational quality. SUMO is freely available. It is available in a standard format, and will probably be available and supported for the coming years. It is difficult to know whether it is used in other projects.

5.4 GUMO+UbisWorld

Physical quality. Different versions of OWL-files are available. Opens in Protégé.

Empirical quality. A visualisation as foldable trees in online browser is available. The full ontologies cannot be viewed, as only smaller parts can be viewed at a time.

Syntactic quality. The OWL validator classifies GUMO+UbisWorld to OWL Full.

Semantic quality. Considering the general description we would expect several similarities with what we need in relation to a person profile. However, concepts we would think of as relations are modelled as classes, independently of the person class. In GUMO, classes that describe a person and his surroundings are modelled, e.g. *DomainDependentDimensions* with subclass *Interest*, and *BasicUserDimensions* with the subclass *Demographics*. In the UbisWorld extension, a *Person* class is included. Hundreds of Person instances are defined, but do not have properties. Few properties are modelled, and none are relevant for us. From this we find that there is a mismatch in how the domain and the related concepts are modelled. Therefore, personal information, stable interests, and temporary interests can not be fully modelled.

Pragmatic quality. Several OWL versions of GUMO and UbisWorld exist, but when to use which one is not specified. The Web sites do not provide any tutorial or user manual. The online ontology browser provides foldable trees of selected parts, together with elements and statements about them in a separate window. It is unfortunate that the browser in the tree structure includes symbols that are not explained (e.g. grey/orange squares, auxiliaries, ranges), and that the assumed corresponding concepts in the OWL versions are modelled as classes. Information presented in the browser contains different, sometimes more, knowledge and different representation of concepts than the OWL versions. This inconsistency between browser and OWL versions is confusing.

Social quality. GUMO is still under development. It is difficult to know how many followers there are, but several papers about GUMO and the environment have been published [52],[53] from a group of people related to its development.

Organisational quality. Need a user profile and password to access available files, but freely available once logged in. It is available in a standard format, and will probably be available and supported the coming years. Might not be stable.

5.5 Summary of Evaluation

The result of the evaluation is summarised in Table 1. We see that for physical quality all providers have available OWL files for download. Even the knowledge bases that

have not initially been created as OWL-ontologies have been translated into OWL. All can be viewed and edited in Protégé except OpenCyc.

Only SUMO provides a visual view of the structure of the ontology, hence giving an idea of content and class relationships. Empirical quality for FOAF is also ok since it is the smallest ontology, and therefore has a structure that can be understood without visualisations. Even though GUMO+UbisWorld provide a foldable tree structure, it only gives a partial view and is inconvenient to use. OpenCyc and concept browsers do not give much insight on overall structure.

When it comes to semantic quality we find differing. The large ontologies have a wider view on a person than what is useful for our purpose. We view a person as a physical actor that needs to be described so that he can act in the world. The personal information category of the profile has been best covered by FOAF and SUMO. Stable and temporary interests have not been covered completely by any of the ontologies. GUMO+UbisWorld have modelled many relevant aspects of a person, but they are not connected to a person class, therefore causing a semantic mismatch relative to our specification. Also FOAF and GUMO+UbisWorld have irrelevant elements, but not to the same extent as the large ontologies.

Table 1. Summary of evaluation

Quality category	FOAF	OpenCyc	SUMO	GUMO+ UbisWorld
Physical	Available	Available, but too big to open	Available	Available
Empirical	Ok	Less satisfactory	Ok	Less satisfactory
Syntactic	OWL Full	Not decidable	OWL Full	OWL Full
Semantic	Partial overlap but not complete, ok validity	Difficult to decide relevancy, poor validity	Partial overlap but not complete, poor validity	Overlap, but modelling mismatch
Pragmatic	Ok	Not satisfactory	Not satisfactory	Not satisfactory
Social	Mature and widely used	Assumed mature, not specified how much it is used	Assumed mature, not specified how much used it is	Not mature, but referenced
Organi-sational	Free, accessible, and stable	Free, not accessible, and probably stable	Free, accessible, and probably stable	Free, accessible, and not stable

Unfortunately it is difficult to understand the logic behind the structure of the ontologies. In general, the models are difficult to read and there are few practical examples where they are used. Hence, the pragmatic quality for particularly OpenCyc, SUMO and GUMO+UbisWorld is poor. Browsers that describe separate concepts are not sufficient to understand the model as whole, and how the different fragments are connected. More documentation directly related to the ontologies and explanations would be advantageous. FOAF has the best pragmatic quality.

The social quality also differs. FOAF is the only ontology explicitly stated as stable, while OpenCyc and SUMO are based on stable knowledge bases. However, whether the OWL versions themselves are prone to changes is not mentioned. FOAF is the most mature ontology, based on its use and number of adaptors. There are small

variances regarding the organisational quality of the ontologies. All ontologies are freely available and are likely to be available in the future. However, with differing indications for how stable they will be.

6 Conclusions

In this paper we have developed an approach to evaluate reuse potential of ontologies. The approach consists of the use of a specialisation of a generic quality framework, SEQUAL, which we have developed and used for the evaluation of FOAF, OpenCyc, GUMO+UbisWorld and SUMO. Reusable ontologies have several potential benefits. Most important, reuse of an existing model is advantageous to save time and resources. In addition, evolution in a later stage benefits from reused ontologies as they would support development based on a shared terminology and understanding that has already been used.

The result of the evaluation is that reuse of these ontologies will not be straight forward. None of the ontologies satisfy the majority of quality requirements. Therefore, it will also be time consuming if parts of any of the ontologies were to be built on. As a result, the ontology has been created from scratch using OWL DL.

In addition to being useful for evaluating ontologies for a particular domain (here profile ontology), the approach gives general insight and information about the evaluated ontologies and about reuse of ontologies in general. It seems like the evaluated ontologies are not made to be easily reusable. Particularly the creation of ontological versions in OWL of existing knowledge bases seems not well thought-through. In the first place, they are created in OWL Full, which gives no computational guarantee. Second, little documentation is available for the modelling decisions and how to use and understand the ontologies.

Our evaluation of reuse of these ontologies is in accordance with the two "rules of three" in software development introduced by Glass [54]: "It is three times as difficult to build reusable components as single use components, and a reusable component should be tried out in three different applications before it will be sufficiently general to accept into a reuse library". This contradicts the purpose of ontologies enabling understanding and reuse. When an ontology has been created for a specific purpose by a set of modellers it is shared between them. A different set of modellers would probably have a different view of the world. For an ontology to be reusable more effort is needed in the construction, and it should be used in several applications.

The created profile ontology is applied in an implementation using OWL API [55] and the reasoner Pellet [56] for access and manipulation of ontologies. The implementation will be evaluated according to developed personas and scenarios. In addition, the personalisation concept will be tested using mock-ups with test people through the RECORD Living Lab [57].

References

1. Berners-Lee, T., Handler, J., Lassila, O.: The Semantic Web. Scientific American (May 2001)
2. Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5 (1993)

3. Uschold, M., Gruninger, M.: Ontologies: Principles, methods and applications. Knowledge Engineering Review 11 (1996)
4. Noy, N.F., McGuiness, D.L.: Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880 (March 2001)
5. Heckmann, D.: Ubiquitous User Modeling. PhD Thesis, Vol. PhD Thesis. Saarland University, Germany (2005)
6. Strasunskas, D., Tomassen, S.L.: Empirical Insights on a Value of Ontology Quality in Ontology-Driven Web Search. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part II. LNCS, vol. 5332. Springer, Heidelberg (2008)
7. Krogstie, J.: Integrated goal, data and process modelling: from TEMPORA to model-generated work-places. In: Johannesson, P., Søderstrøm, E. (eds.) Information Systems Engineering: From Data Analysis to Process Networks. IGI Publishing (2008)
8. Lin, Y., Sampson, J., Hakkarainen, S.: An Evalution of UML and OWL using a semiotic quality framework. In: Advanced Topics in Database Research, vol. 4. Idea Group Publishing, Hershey (2004)
9. Su, X., Ilebrekke, L.: A Comparative Study of Ontology Languages and Tools. In: Pidduck, A.B., Mylopoulos, J., Woo, C.C., Ozsu, M.T. (eds.) CAiSE 2002. LNCS, vol. 2348, p. 761. Springer, Heidelberg (2002)
10. Krogstie, J.: Evaluating UML Using a Generic Quality Framework. In: Favre, L. (ed.) UML and the Unified Process, pp. 1–22. IRM Press (2003)
11. Krogstie, J.: A Semiotic Approach to Quality in Requirements Specifications. In: Working Conference on Organizational Semiotics. Proceedings og IFIP 8.1, Montreal, Canada (2001)
12. Hakkarainen, S., Strasunskas, D., Hella, L., Tuxen, S.: Choosing Appropriate Method Guidelines for Web-Ontology Building. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, pp. 270–287. Springer, Heidelberg (2005)
13. Hakkarainen, S., Strasunskas, D., Hella, L., Tuxen, S.: Classification as Evaluation: A Framework Tailored for Ontology Building Methods. In: Siau, K. (ed.) Advanced Topics in Database Research Series, vol. 5, pp. 41–62 (2006)
14. Krogstie, J., Sindre, G., Jørgensen, H.: Process Models as Knowledge for Action: A Revised Quality Framework. European Journal of Information Systems 15, 91–102 (2006)
15. Bonnet, S.: Model Driven Software Personalization. In: Smart Objects Conference (SOC 2003), Grenoble, France (2003)
16. Farshchian, B.: Daidalos, Response to recommendation number 6 from audit 2005, Research questions and achievements of WP4 (2005)
17. Zimmermann, A., Specht, M., Lorenz, A.: Personalization and Context Management. User Modeling and User-Adapted Interaction, August 2005, vol. 15, p. 28. Springer, Netherlands (2005)
18. Delivery Context Ontology W3C Working Draft (June 16, 2009), `http://www.w3.org/TR/dcontology/`
19. Dey, A.K., Abowd, G.D.: Towards a Better Understanding of Context and Context-Awareness. In: The Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000). The Hague, The Netherlands (2000)
20. Protégé Ontology Editor and Knowledge Acquisitions System, `http://protege.stanford.edu/`

21. WonderWeb OWL Ontology Validator, http://www.mygrid.org.uk/OWL/Validator
22. Chandrasekaran, B., John, R.J., Benjamins, V.R.: What Are Ontologies, and Why Do We Need Them? IEEE Intelligent Systems 14, 20–26 (1999)
23. Gandon, F.L., Sadeh, N.M.: A Semantic E-Wallet to Reconcile Privacy and Context Awareness. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 385–401. Springer, Heidelberg (2003)
24. Gandon, F.L., Sadeh, N.M.: Semantic Web Technologies to Reconcile Privacy and Context Awareness. Journal of Web Semantics 1, 27 (2004)
25. Mendis, V.: Rdf user profiles - bringing semantic web capabilities to next generation networks and services. In: Proceedings of the ICIN Conference (2007)
26. Stan, J., Egyed-Zsigmond, E., Joly, A., Maret, P.: A User Profile Ontology For Situation-Aware Social Networking. In: 3rd Workshop on Artificial Intelligence Techniques for Ambient Intelligence (2008)
27. Ghosh, R., Dekhil, M.: Mashups for semantic user profiles. In: Proceeding of the 17th international conference on World Wide Web. ACM, Beijing (2008)
28. FOAF project Web page, http://xmlns.com/foaf/spec
29. FOAF Vocabulary Specification 0.91 (2007), http://xmlns.com/foaf/spec/
30. Introducing FOAF, http://www.foaf-project.org/original-intro
31. Golbeck, J., Rothstein, M.: Linking Social Networks on the Web with FOAF. In: Proceedings of the Twenty-Third Conference on Artificial Intelligence AAAI 2008 (2008)
32. Recent foafnews Bookmarks (02.05.2009), http://delicious.com/tag/foafnews
33. Dodds, L.: An Introduction to FOAF, http://www.xml.com/pub/a/2004/02/04/foaf.html
34. Cycorp, Inc. Web page, http://www.cyc.com/
35. OpenCyc Web page, http://opencyc.org/
36. OpenCyc for the Semantic Web - Searching for OpenCyc Content, http://sw.opencyc.org/
37. OpenCyc concept browser, http://www.cycfoundation.org/concepts
38. OpenCyc blog, http://www.cycfoundation.org/blog
39. SUMO Web Page, http://suo.ieee.org/SUO/SUMO/index.html
40. Niles, I., Pease, A.: Towards a Standard Upper Ontology. In: Proceedings of the 2nd International Conference on Formal Ontology in Information Systems, FOIS-2001 (2001)
41. Standard Upper Ontology Knowledge Interchange Format, http://suo.ieee.org/SUO/KIF/suo-kif.html
42. SUMO Publications, http://www.ontologyportal.org/Pubs.html
43. Miller, G.A.: WordNet: a lexical database for English. Communications of the ACM 38, 39–41 (1995)
44. Miller, G.A., BeckWith, R., Fellbaum, C., Gross, D., Miller, K.: WordNet: An on-line lexical database. International Journal of Lexicography 3, 235–244 (1990)
45. SUMO online browser, http://sigma.ontologyportal.org:4010/sigma/Browse.jsp?kb=SUMO&lang=EnglishLanguage
46. Sevcenko, M.: Online presentation of an upper ontology. In: Proceedings of Znalosti 2003, Ostrava, Czech Republic (2003)
47. SUMO classes, http://www.ontologyportal.org/images/SUMOclasses.gif

48. Sigma Knowledge Engineering Environment, http://sigmakee.sourceforge.net/
49. GUMO Web page, http://gumo.org/
50. Heckmann, D., Schwartz, T., Brandherm, B., Kröner, A.: Decentralized User Modeling with UserML and GUMOHeckmann05.pdf. In: Proceedings of DASUM 2005, Edinburgh, Scotland (2005)
51. UbisWorld Web page, http://www.ubisworld.org/
52. Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., Von Wilamowitz-Moellendorff, M.: GUMO - The General User Model Ontology. In: Ardissono, L., Brna, P., Mitrović, A. (eds.) UM 2005. LNCS (LNAI), vol. 3538, pp. 428–432. Springer, Heidelberg (2005)
53. Heckmann, D.: Integrating privacy aspects into ubiquitous computing: A basic user interface for personalization. In: Proceedings of the AIMS 2003, Artificial Intelligence in Mobile System (2003)
54. Glass, R.L.: Facts and fallacies of software engineering. Addison-Wesley, Reading (2003)
55. The OWL API (2010), http://owlapi.sourceforge.net/
56. Pellet: OWL 2 Reasoner for Java (2010), http://clarkparsia.com/pellet
57. RECORD Living Lab, http://www.recordproject.org/

Distribution of Effort among Software Development Artefacts: An Initial Case Study

Niklas Mellegård and Miroslaw Staron

Department of Applied Information Technology
Chalmers University of Technology, University of Gothenburg
{niklas.mellegard,miroslaw.staron}@ituniv.se

Abstract. Model-driven development aims at increasing productivity by raising the abstraction level of software specifications and introducing automated transformations for replacing lower level specifications. To assess benefits of replacing a legacy development process with a model-driven approach, one needs to establish a baseline of the current process with respect to the effort invested in the development artefacts. In this paper we report on an initial case study in which we investigate the main artefacts in the analysis and design phase with respect to required effort and perceived importance. We studied a non-model driven development of software based automotive functionality and our initial results show that a few artefacts receive the majority of effort, and that the artefacts that receive the most effort are not the most important ones. The initial results indicate that the distribution of effort between models and other artefacts is similar to that of model-driven projects in spite of the project being perceived and characterized as code-centric.

Keywords: Software engineering, Requirements, Analysis, Design, Modelling, Process.

1 Introduction

Model-driven development (MDD) [1-3] has the goal of increasing development productivity and the quality of software based products by raising the level of abstraction at which the development takes place. Several studies have provided evidence showing that the application of MDD in large industrial project has indeed improved productivity and the quality of the products (e.g. [4-7]), while other studies show mixed results and also point out the lack of objective empirical evidence [8].

Industrial software development, however, rarely provides the possibility to go from code-centric to model-driven software development in a single step and to the full extent. One example of software development domains where this is not possible is automotive software development. The domain is characterized by the existence of a lot of legacy software and high interdependence between car manufacturers and suppliers of car components. This high interdependence requires precise specifications that have to provide possibilities for interoperating of software development

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 234–246, 2010.

practices at the manufacturer's side and the supplier's side. This, in consequence, means that a number of practices for using models are used in parallel – e.g. UML, SimuLink. Since modelling notations differ in the degree of formality and quite often can be used interchangeably, it is important to optimize the cost of using these notations. The cost has to be balanced with the benefits that these models bring and perhaps not used in those parts of software development where code-centric approach is already very efficient.

In this case study we explore the costs and efforts of using different modelling notations and different abstraction levels for specifying requirements and designing the software in the automotive domain. The case study was conducted at Volvo Car Corporation (VCC), within the department responsible for electronic and software systems in Volvo automobiles. The research question in the case study was:

> *What is the distribution of the effort invested in the artefacts, documents and deliverable products in the main software specification phase?*

Addressing this research question provided the possibility to assess the costs of using different kinds of modelling notations in software development. The data was collected via document analysis and a sequence of interviews with architects and project managers. The main perspective of the results is the project managers'. The results show that there are a few artefacts which require significantly more effort than other artefacts and that the perception of what is important is different from where the effort is spent.

The rest of the paper is structured as follows. Section 2 describes the most related work in the area of studying effort of software development activities. Section 3 presents the fundaments of the case study – description of the de-facto process of software development at VCC. Section 4 outlines the design of the case study, while section 5 presents the results, and section 6 concludes the paper.

2 Related Work

The concepts which we investigated in this case study were based on the map (referred hereon to as *domain model*) of the development phases reported in our previous research [9]. In [9] we have evaluated the correctness of the domain model, discussed its expressive power and assured that it represents the de-facto development process at VCC.

Mohagheghi and Dehlen [8] conducted a review of documented modelling experiences and concluded that there are few reported results of how the MDD scales to large system development. Furthermore, the paper concludes that there often is a lack of company baselines which results in subjective evaluations. Our case study is intended to contribute to such a company baseline intended to be used to compare the development process with other companies.

Method engineering [10-12] recognizes the fact that no development method will fit every development task. Instead, method engineering aims at establishing a framework for adapting existing methods to better fit the development task at hand [11]. However, what is entailed in "better fit" differs greatly between projects. In this case study we examine – from the perspective of project managers – which development artefacts are

considered central to the development process and thereby contributing with evidence of which parts of the development process should be in focus if an adaption of the current process be made.

In [7] we contrasted how effort was distributed between the different development phases in model-driven and code-centric projects within a company in the telecommunication domain. The results showed that the overall efficiency of the development process improved by adopting an MDD approach, the evidence for the analysis and design phase were not conclusive. In this case study we examine the analysis and design phase, specifically with respect to the development artefacts created. Furthermore, in [7] we found that the implementation phase had the most improvement potential; however, in the automotive domain the majority of implementation is done by third-party suppliers. This raises the question of how an MDD approach can be used to improve the efficiency of the analysis and design phase. In this case study we have taken the initial step by examining the distribution of effort among and the perceived importance of the artefact developed in the phase.

Bollain and Garbajosa proposed in [13] an extension to the ISO/IEC 24744 [14] meta-methodology standard. Their purpose was to extend the standard in order to better fit a document-centric (referred to as *code-centric* in this paper) development process. Our research has similar goals, but instead focuses on the use of models within a non-MDD development process. The case study reported in this paper provides evidence of which artefacts are the central ones, which in turn indicate where the focus of a modelling meta-model may be.

Broy [15] outlines the challenges in automotive software engineering, and also outlines a structural view on the development process, which he calls a comprehensive architecture. Our case study focuses on one particular part of this architecture, namely what Broy refers to as the *Design level* (which we refer to as the *function definition phase*). Moreover, Broy concludes that although models are used throughout the automotive software development process, their use is fragmented. This means that the benefit of having a coherent model chain – such as automatic artefact generation and traceability – is lost. Our case study contributes with empirical evidence that characterizes the development process – with regard to effort and importance of the constituent artefacts – which we hope will contribute with further evidence of how such an integrated modelling chain can be created in an optimal way.

Heijstek and Chaudron [5] report on an empirical study regarding model size, complexity and effort in a large model driven process, but whereas their study included the investigation of effort distribution among categories of development activities for the whole development process, our study focuses on how effort is distributed among the artefacts produced in one of the development activities, and specifically on the effort distribution among types of models and requirements. Moreover, Heijstek and Chaudron report on the importance and centrality of models in a pure MDD project, whereas our case study is conducted at a company which does not use an MDD approach. We elaborate on this in section 5.

3 Domain Model

The development of software based functions at VCC makes extensive use of models, although it cannot be classified as 'model-based' according to the definitions by

Brown [3]. The types of models used are both descriptive and prescriptive, as defined by Ludewig [16]. However the use of modelling techniques is fragmented and does not constitute a coherent chain of models, as model-driven development advocates [9]. This fragmentation is typical and evidence for such a fragmentation in new adopters of model driven engineering has also been reported by Broy [15].

The development of the products at the electrical department is separated into four phases (described in more detail in our previous research [9]) which – as shown in Fig. 1. – are: the *strategic phase* which is focused on selecting a number of high level features; the *function definition phase* in which individual functions are analysed and specified, and finally; the *system* and *node development phases* which are concerned with designing and specifying a software and hardware solution that realize the functions. The strategic phase is concerned with product planning, where a set of high-level features are selected in order to make the vehicle model competitive at the market. The subsequent phases contain the engineering activities which are of main interest in our research, and the activities in the function definition phase are the focus of this case study.

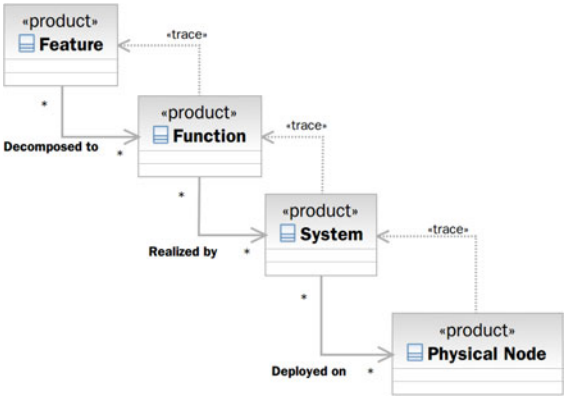


Fig. 1. Product Development Stages

The activities in the function development phase result in design and specification of a function, e.g. airbags or collision warning. In the phase, only the implementation independent properties are under consideration, such as desired behaviour and proposed algorithmic solutions.

In the system development phase the designers specify a system solution that satisfies the specified behaviour of the function on a specific hardware and software platform. In the component development phase the focus is on design and specification of the physical components that are part of the system solution. The components are then implemented, mostly by third party suppliers. Since the implementation is done by the suppliers, the most important design phase is the function design phase, which we study in our research.

Fig. 2. shows the key concepts we have identified in the function development phase [9]. The map in Fig. 2. was created using domain modelling in UML (i.e. using class diagrams with classes as concepts, associations and dependencies). We used three stereotypes to distinguish between types of elements in the domain model:

- *Product*: elements which are results (or in early stages of the project – prospected results) of the software project – e.g. features, functions, components.
- *Artefact*: elements which are used in the development process – e.g. models, requirements, abstract pseudo-code, algorithm descriptions

- *Document*: elements which are official documents prescribed by the software development process at VCC and can be deliverables from the process – e.g. requirement specification

The above three categories show that the three kinds of elements interact and complement each other.

The concepts we have identified in the function development phase include

- The function requirements specification
- Description of the functions behaviour from an end-user perspective, shown in Fig. 2. as *Use Cases*, *Specification Model* and *Requirement*
- Descriptions of algorithmic solutions, shown in Fig. 2. as *Simulation Model* and a textual description of the simulation model, shown as *Design Description*
- An initial design, shown in Fig. 2. as *Logical Design*

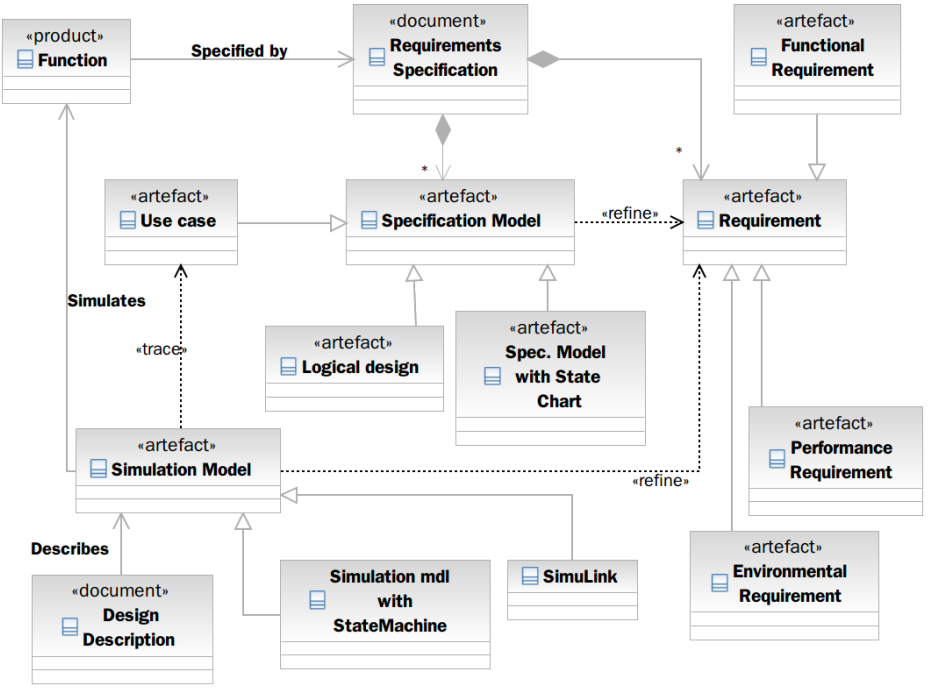


Fig. 2. Artefacts, documents, and software products in the function development phase

4 Design of the Case Study

When designing this case study we followed the case study design proposed by Yin [17]. Initially a case study protocol was set up in which two case study propositions were defined. For each of the two propositions, a hypothesis was formulated and interview questions created that would elicit evidence to support or refute the hypothesis. Using the data collected from the interviews we evaluated the hypotheses and addressed our research question.

In the following subsections we present an outline of the design of the case study: propositions, units of analysis, data collection procedures, and criteria for interpreting the data.

4.1 Proposition 1 – Relative Importance of Artefacts

Based on our previous research [9, 18] we anticipated that there would be large differences between the relative importance of artefacts and the effort put into their development. In other words we anticipated that the most effort-intensive artefacts were probably not the most important ones from the perspective of product development. These differences meant that there could be a 'waste' of effort (in terms of LEAN [19]) and therefore there was a room for improvement or effort optimization.

4.2 Proposition 2 – Effort Distribution among Artefacts

We proposed that there would be a significant difference between the amounts of effort spent on the artefacts identified in the domain model. Furthermore, we proposed that there is a pattern (with respect to artefact type) in how the effort is distributed among the development artefacts.

In order to identify improvement opportunities in the current process, it is important to know which artefacts are the most effort-intensive and therefore which artefacts we should concentrate on in the first place. If the perceived effort and importance do not match, then there is a need for deeper investigation and optimization of the development process at the company.

4.3 Units of Analysis

The main unit of analysis in this case study was the importance of and the relative amounts of effort invested in the development artefacts, specifically requirements, different types of models and documents. However, there was also an embedded unit of analysis in this case, namely the role of the interviewed subjects, as each role has a particular perspective on both the importance of the artefacts in question as well as the amount of effort that is invested in creating them.

4.4 Data Collection Procedures

The data collection consisted of structured interviews in which we collected evidence of the perceived importance and effort spent on the artefacts we identified as being the key concepts in the function development phase (see Fig. 2.).

We first presented and described domain model showing the key concepts in the function development phase (Fig. 2.). The purpose with this presentation was to provide the respondent with the context of the study and also to provide the opportunity to ask about and comment on the content and structure of the domain model (e.g. add concepts that were found important or change relationships among the shown concepts). We conducted this part as a focused interview [17], and in order to ensure construct validity, we noted the comments and incorporated them in the following parts of the interview (in accordance with Yin [17]).

We primarily wanted to interview subjects that had an overview of the entire project in order to avoid introducing bias to specific activities; therefore our sample consisted of:

- **Main study.** The main study sample consisted of three project managers. All subjects had been working at VCC for many years, and within the function development phase between 6 months and several years
- **Pilot study.** The sample used in the pilot study consisted of two architects who had been working as architects for several years.

4.4.1 Units of Analysis for Proposition 1

The unit of analysis in proposition 1 was the perceived importance of the artefacts in the domain model. In order to collect data regarding their importance, we divided the artefacts in the categories models, requirements and documents, and asked the interviewee which they thought was the most important to get right in order to ensure the success of developing the function. Furthermore, in order to contribute to our understanding of why the particular category of artefacts was considered the most important, we asked to interviewee to explain and give examples. These comments were noted for further analysis.

4.4.2 Units of Analysis for Proposition 2

Regarding proposition 2, we collected data by applying a variation of the $100 test technique [20], where we asked the interviewees to place an appropriate amount of money on each of the artefacts in the domain model. Because the artefacts shown in the domain model are constituent parts in the document, we considered effort spent on the documents to be editorial work or tasks not associated with the other artefacts shown in the domain model.

The amount of effort assigned to each artefact was normalised by calculating for each concept the percentage of the total amount distributed. In addition, the sum of effort for each category of artefacts was also calculated (in order to match the categories of artefacts from proposition 1).

We followed up on the result of the $100 test by asking the interviewees whether they thought it was straight-forward to assign the amount to each artefact, and if not, what they felt was the main difficulties in doing so. Furthermore, we asked whether they thought that their distribution of effort would be representative for other roles in their project, as well as for other projects.

4.5 Interpreting the Findings

We intended to analyse the results by means of pattern matching [17]. Based on our earlier research, we anticipated that there could be a difference between which artefacts were considered the most important and the effort that was put into their development. By comparing the results of propositions 1 and 2, we examined whether the category of artefacts considered most important was in fact the same as the one most effort was invested in.

By studying the effort distribution at VCC we could establish a reference point for comparing effort distribution in MDD projects in different companies or domains – e.g. by comparing that to the study conducted at Ericsson [7].

4.6 Validity Evaluation

We have identified and grouped the threats to validity in our study according to recommendations of Yin [17]:

Construct validity. The main instrument of this study is the domain model around which the questions of the interview revolved. In order to ensure construct validity, we began each interview session by presenting the domain model and asking whether the interviewee found any concepts missing. In neither of the interviews did we identify any issues with the domain model; we thus consider the model to be complete and correct.

Internal validity. As any interview study we anticipated some personal bias in the answers from the interviewees. In order to minimize this threat we triangulated the results by conducting document analysis and a pilot study with architects who are not involved in the actual case study.

External validity There is a risk that the results are too specific to Volvo Car Corporation. In order to validate the results we plan to conduct a similar study at other companies which our university collaborates with.

Reliability. As part of the case study design, we have created a case study protocol which ensured that we conducted the study and collected the data in a consistent manner. By using this protocol, we believe that the study can be reliably reproduced.

5 Results

5.1 Pilot Study Results

In the pilot study we interviewed 2 architects. The subjects, however, were not working with detailed design of vehicle functions and were therefore used only to validate the design of our case study.

As a result of the pilot study we found that the domain model was valid, i.e. complete and correct. None of the architects identified issues (such as missing or invalid concepts) with the domain model (nor did such issues arise during the rest of the interviews with project managers). Furthermore, as result of the pilot study we preliminary identified which artefacts might be the most important ones – e.g. Use Cases. This information made us aware when interpreting later findings in the case study.

We have also found that it is probably the models that contain most of the information, but that this is dependent on the role of the subject. As a result of this remark we have checked the list of planned subjects in our study and found no risk of introducing bias by missing important roles.

Finally we have found that specification models and the related requirements are more important than simulation models, although it might vary significantly depending on the product that is under development. As a result of this we have ensured that we introduced another question – which products the interviewee has in mind when answering our questions.

5.2 Case Study Results

In this section we report the initial results of the case study. At this point three project managers have been interviewed; our study, however, is ongoing and results from a larger sample will be reported in a future paper.

5.2.1 Case Study Proposition 1

With regard to the importance of the development artefacts, we found that requirements, and especially performance and functional requirements related to design components such as sensors and actuators, were considered the most important ones. The rationale explaining this was, according to the interviewees, that other models in the design of the functional solution, such as the simulation models, rely on these requirements as assumptions about the available design components and their properties. Moreover, we found that the simulation models were not considered to be that important, as they can easily be reworked if the requirements change.

We had initially anticipated that models would be the most important artefacts, as they are generally more expressive than text. However, we found that it is rather the level of detail and the need to express requirements unambiguously that is considered most important from the project manager's perspective. We found that the models are, from the project manager's perspective, mainly seen as support for the requirements; specification models are used to provide structure to the requirements or to provide them with context. In particular, the simulation models are used to explore possible functional solutions in order to learn to understand what the detailed requirements are.

Furthermore, the detailed requirements created in the function definition phase are provided as input requirements to the subsequent system and component development phases, where they are further refined to form part of the specification that is eventually provided to the third-party supplier. Therefore it was considered highly important that the requirements were fully understood and correct.

5.2.2 Case Study Proposition 2

The models used in the function definition phase, listed in Table 1, consists of

- One or a few use case descriptions, including alternative flows and error states;
- Specification models, such as state machines, refining the use-case into a detailed functional specification;
- Simulation models intended both as validate the function design, as well as to serve as a tool for the function designer to learn to understand the relevant requirements.

The level of formality of the models ranges from informal, such as use cases, to formal ones such as executable simulation models.

From the normalized result of the $100 technique, shown in , we found that about twice the effort is invested in models compared to requirements. This initial finding supported our anticipated result that the artefacts considered most important are not the ones the most effort was spent on. Furthermore, we found that the artefact the most effort is spent on is the simulation models which can be explained by the fact that these models are used to make the requirements more precise.

We also found that a significant amount of effort (approximately the same as is spent on requirements) is spent on editing documents.

Table 1. Models in the Function Definition Phase

Artefact	Model	Notation
Function Description	Use-case	UML/Use-case
	Specification models Logical design State charts	 UML/Class Diagram UML/State Machine
	Simulation models Simulink Statemate Stateflow Powerpoint Video sequences	 Proprietary Proprietary Proprietary N/A N/A

When showing the results to the interviewees and their managers after the study, their reaction was that this was against their expectations – they expected more effort to be spent on the requirements than it is now. This called for a subsequent (planned for future) study about detailed optimization techniques how to improve the throughput of the process. Possible targets for optimization is the multitude of different modelling techniques and notations used that are compatible to a limited degree. Similar observations and a report on how such challenges was overcome when Motorola successfully introduced MDD in their process is reported in [4].

Table 2. Effort distribution in percent per artefact as estimated by the project managers

Artefact	PM 1	PM 2	PM 3[1]
Models	*(56.4)*	*(50)*	*(73.4)*
Simulation model	39.7	30	26.7
Use cases	7.9	20	20
State Machines / other specification models except UCs	4.0		26.7
Logical Design	4.8		
Requirements	*(23.8)*	*(30)*	*(26.7)*
Requirements (all types)	23.8	30	26.7
Documents	*(19.8)*	*(20)*	
Design Description	11.9		
Requirements Specification	7.9		

5.3 Interpretation of the Results

Interestingly, from the result of the case study propositions we found that the artefacts which were considered as the most important were not the same artefacts that receive the most effort.

Whereas requirements were considered very important, they received about half the effort compared to the simulation models.

Furthermore, creating the documents – mainly considered by the interviewees to be editorial work – required almost as much effort as the requirements do, while not considered as important. We raised an improvement potential here – perhaps the tedious work should be reduced.

We have found that the simulations models are often used during the function definition phase to explore different functional solutions, and thereby assisting the developers in understanding the requirements (conf. [9]). This would explain why such large amount of effort is spent on developing these models. However, this does not explain why the simulation models are not considered as important as the requirements.

Furthermore, the results shown in indicate a strong correlation between level of model formality and amount of effort required. Developing the formal specification model takes according to one subject almost all modelling time – approx. 5 times more than the next one – Use cases model.

As the formal simulation models are not considered as important to the success of the product, it raises the question of why their importance is perceived to be so low. We plan further investigations at the company to explore it to a deeper extent.

The results of case study proposition 2 indicate that documents require a substantial amount of effort and as the majority of their contents are compositions of the other artefacts. Here automating the process of document creation may be a way to improve process efficiency.

Although VCC cannot be considered as using an MDD approach, we have found that the relative amount of effort spent on models compared to other artefacts is similar to what Heijstek and Chaudron found in their paper [5] where they investigate pure MDD projects. In their study, Heijstek and Chaudron found that in a pure MDD project at an IT service provider, 59% of all effort is spent on developing models, whereas in our

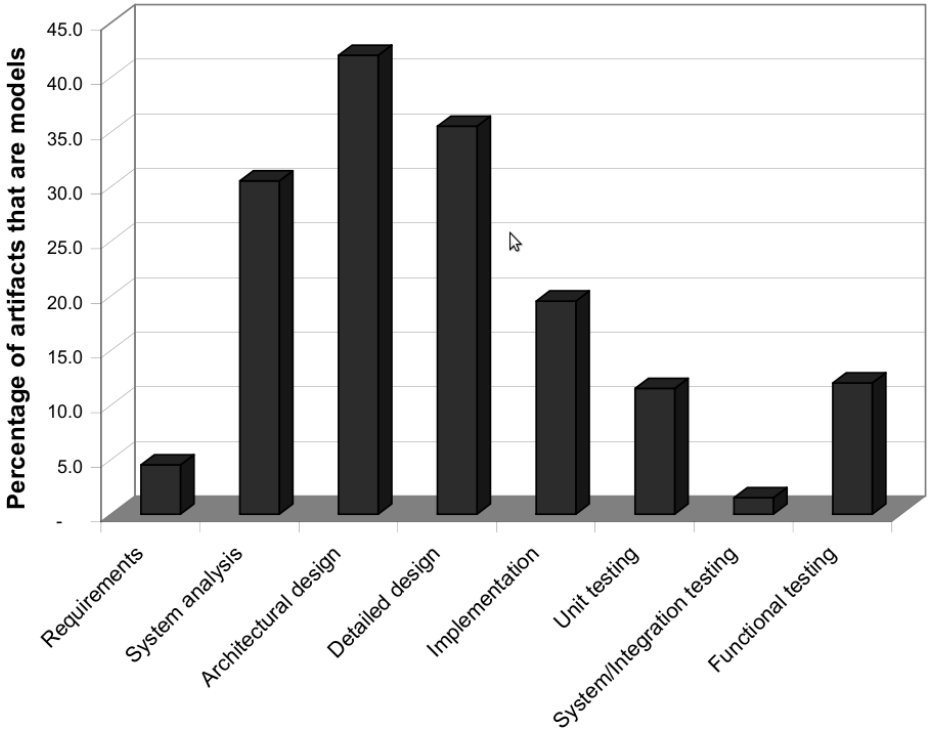


Fig. 3. Percentage of artefacts which contain models per phase from [7]

study we found it to be about 60%. Although the product development process at VCC does not comply to a pure MDD approach – but can rather be described as document-centric [21, 13] – models are frequently used in the requirements specification documents, but they are mainly used to provide context and structure for the requirements.

Furthermore, Heijstek and Chaudron found quantitative evidence that some kinds of models receive more attention than others. In our case study we found similar qualitative results. This finding suggests that a strategy for improving process efficiency is to concentrate such improvement efforts on particular types of models and their associated activities. The results of our study indicate that such effort should be directed at simulation models and their traceability to the requirements that constrains their design.

In the light of our previous research [7] we have investigated how much modelling is present in different phases – the results which are presented in Fig. 3. The results show that modelling is rather a large part of the phases covered by our study – analysis, architectural design and system design – which are all part of the function development phase. Our new findings raise an important question – how much of the modelling in these phases is essentially needed?

6 Conclusion

The objective of this study was to investigate how effort and the perceived importance are distributed among artefacts identified in the function development phase of software based vehicle functions. In this paper we have reported on an initial case study in which we interviewed project managers responsible for a software based safety and security related functions.

Our initial results show that the artefacts that are considered the most important are not the ones that the most effort is invested in. While detailed textual requirements are considered the most important artefact in order to ensure a successful product, it is executable simulation models that receive the majority of effort during development. Furthermore, from the interview we found that the simulation models are not considered very important as they are easy to modify if the requirements should change.

Moreover, we have in this initial study of a non-MDD project found that the amount of the total development effort invested in models is similar to the pure MDD project reported in [5] – about 60% in our case and 59% in the MDD case.

As part of our continuing work we are conducting this case study on a larger scale by interviewing more people in a number of different roles in order to add to the empirical evidence of how effort and importance of development artefacts such as models and requirements are distributed.

Acknowledgements

This research is partially sponsored by The Swedish Governmental Agency for Innovative Systems (VINNOVA) under the Intelligent Vehicle Safety Systems (IVSS) programme.

References

1. Atkinson, C., Kuhne, T.: Model-driven development: a metamodeling foundation. IEEE Sw. 20, 36–41 (2003)
2. Brown, A.: An introduction to Model Driven Architecture - Part I: MDA and today's systems, http://www.ibm.com/developerworks/rational/library/3100.html
3. Brown, A.W.: Model driven arch.: Principles and practice. Sw. and Sys. Model 3, 314–327 (2004)
4. Weigert, T., Weil, F., Marth, K., Baker, P., Jervis, C., Dietz, P., Gui, Y., van den Berg, A., Fleer, K., Nelson, D., Wells, M., Mastenbrook, B.: Experiences in Deploying Model-Driven Engineering. In: Gaudin, E., Najm, E., Reed, R. (eds.) SDL 2007. LNCS, vol. 4745, pp. 35–53. Springer, Heidelberg (2007)
5. Heijstek, W., Chaudron, M.: Empirical Investigations of Model Size, Complexity and Effort in a Large Scale, Distributed Model Driven Development Process. In: 35th Euromicro Conference on Software Engineering and Advanced Applications, 2009. SEAA 2009, pp. 113–120 (2009)
6. Baker, P., Loh, S., Weil, F.: Model-Driven Engineering in a Large Industrial Context — Motorola Case Study. In: Briand, L.C., Williams, C. (eds.) MoDELS 2005. LNCS, vol. 3713, pp. 476–491. Springer, Heidelberg (2005)
7. Staron, M.: Transitioning from code-centric to model-driven industrial projects – empirical studies in industry and academia. In: Model Driven Software Development: Integrating Quality Assurance. Information Science Reference, pp. 236–262 (2008)
8. Mohagheghi, P., Dehlen, V.: Where Is the Proof? - A Review of Experiences from Applying MDE in Industry. In: Schieferdecker, I., Hartman, A. (eds.) ECMDA-FA 2008. LNCS, vol. 5095, pp. 432–443. Springer, Heidelberg (2008)
9. Mellegård, N., Staron, M.: Use of Models in Automotive Software Development: A Case Study. Presented at the First Workshop on Model Based Engineering for Embedded Systems Design, Dresden, Germany, March 12 (2010)
10. Brinkkemper, S.: Method engineering: engineering of information systems development methods and tools. Information and Software Technology 38, 275–280 (1996)
11. Rolland, C.: Method eng.: towards methods as services. Sw. Process: Improvement and Practice 14, 143–164 (2009)
12. Karlsson, F.: Meta - Method for Method Configuration: A Rational Unified Process Case, dissertation, Linköping University of Management and Engineering, VITS-Development of Informations Systems and Work Context (2002)
13. Bollain, M., Garbajosa, J.: A Metamodel for Defining Development Methodologies. Software and Data Technologies, 414–425 (2009)
14. ISO - International Organization for Standardization: ISO/IEC 24744:2007, software Engineering – Metamodel for Development Methodologies (2007)
15. Broy, M.: Challenges in automotive software engineering. In: Proceedings of the 28th international conference on Software engineering, pp. 33–42. ACM, Shanghai (2006)
16. Ludewig, J.: Models in software eng. – an introduction. Sw. and Sys. Modeling 2, 5–14 (2003)
17. Yin, R.K.: Case Study Research: Design and Methods, 3rd edn. Applied Social Research Methods Series, vol. 5. Sage Publications, Inc., Thousand Oaks (2002)
18. Mellegård, N., Staron, M.: A Domain Specific Modelling Language for Specifying and Visualizing Requirements. In: The First Int. Workshop on Domain Engineering. DE@CAiSE, Amsterdam (2009)
19. Poppendieck, M.: Lean Software Development. In: Companion to the proceedings of the 29th International Conference on Software Engineering, pp. 165–166. IEEE Computer Society, Los Alamitos (2007)
20. Leffingwell, D., Widrig, D.: Managing software requirements. Addison-Wesley, Reading (2003)
21. Rausch, A., Bartelt, C., Ternité, T., Kuhrmann, M.: The V-Modell XT Applied–Model-Driven and Document-Centric Development. In: 3rd World Congress for Software Quality (2005)

FORML 2

Terry Halpin[1] and Jan Pieter Wijbenga[2]

[1] LogicBlox, Australia and INTI Education Group, Malaysia
[2] University of Groningen, The Netherlands
terry.halpin@logicblox.com, J.P.Wijbenga@student.rug.nl

Abstract. A conceptual schema of an information system specifies the fact structures of interest as well as the business rules that apply to the business domain being modeled. These rules, which may be complex, are best validated with subject matter experts, since they best understand the business domain. In practice, business domain experts often lack expertise in the technical languages used by modelers to capture or query the information model. Controlled natural languages offer a potential solution to this problem, by allowing business experts to validate models and queries expressed in language they understand, while still being executable, with automated generation of implementation code. This paper describes FORML 2, a controlled natural language based on ORM 2 (second generation Object-Role Modeling), featuring rich expressive power, intelligibility, and semantic stability. Design guidelines are discussed, as well as a prototype implemented as an extension to the open source NORMA (Natural ORM Architect) tool.

1 Introduction

A conceptual information model includes a conceptual schema as well as a population (set of instances that conform to the schema). Ideally, a conceptual schema specifies the fact structures of interest as well as the applicable business rules in terms of concepts that are intelligible to the business users. Business rules are constraints or derivation rules that apply to the relevant business domain. Alethic constraints restrict the possible states or state transitions of fact populations. Deontic constraints are obligations that restrict the permitted states or state transitions of fact populations. Derivation rules enable some facts to be derived from others.

Business rules may be complex (since the domain being modeled may itself be complex), and are best validated with subject matter experts, who best understand the business domain. In practice, business domain experts may lack the technical expertise required to understand the technical languages used by modelers to capture or query the information model. These languages may be graphical (e.g. class diagrams in the Unified Modeling language (UML) [29]), or textual (e.g. the Object Constraint Language (OCL) [30, 37] supplement to UML). *Controlled natural languages* (unambiguous subsets of natural languages with restricted grammar and vocabulary) offer a potential solution to his problem, by allowing business experts to validate models and queries expressed in language they understand, while the models/queries can still be executable, with automated generation of implementation code.

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 247–260, 2010.

In *fact-oriented modeling* approaches, all facts are treated as instances of fact types, which may be existential (e.g. Patient exists) or elementary (e.g. Patient smokes, Patient is allergic to Drug). In attribute-based approaches such as Entity Relationship modeling (ER) [7] and UML's class diagramming technique, facts may be instances of attributes (e.g. Patient.isSmoker) or relationship types (e.g. Patient is allergic to Drug).

Fact-oriented modeling approaches include *Object-Role Modeling* (*ORM*) [18], Cognition-enhanced Natural Information Analysis Method (CogNIAM) [28], the Predicator Set Model (PSM) [25], and Fully-Communication Oriented Information Modeling (FCO-IM) [1]. The Semantics of Business Vocabulary and Business Rules (SBVR) initiative is also fact-based in its use of attribute-free constructs [31]. An overview of fact-oriented modeling approaches, including history and research directions, may be found in [16].

This paper discusses *Fact-Oriented Modeling Language version 2* (*FORML 2*), a controlled natural language based on second generation ORM (ORM 2) [13]. An introduction to ORM may be found in [14, 18], a thorough treatment in [22], and a comparison with UML in [17]. FORML 2 is a formal yet intelligible textual language with rich expressive power and high semantic stability. The body of this paper discusses its main features and design guidelines, as well as a prototype implementation of FORML 2 as an extension to the Natural ORM Architect (NORMA) tool.

The rest of this paper is structured as follows. Section 2 briefly overviews related work on high level textual languages, both within and outside the fact-oriented community, as well as providing some background on ORM. Section 3 provides a brief overview of the NORMA tool, and its modes of support for FORML 2. Section 4 discusses some of the underlying algorithms and design guidelines. Section 5 provides details of the formal grammar and implementation architecture used to support FORML 2 as an input language. Section 6 summarizes the main contributions and outlines future research directions.

2 Background and Related Research

Provision of a high level, executable rule language based on a formal subset of natural language that is intelligible to ordinary business users has the potential to revolutionize the way software systems are developed. It is not surprising therefore, that many attempts have been made, and are being made, to achieve this goal. The first language of this nature, Reference and Idea Language (RIDL) [27], was developed in the 1980s based on an early version of NIAM; the model declaration part was implemented in the RIDL* tool, but relationships were restricted to binaries, and the query part was never implemented.

Following RIDL, other fact-oriented modeling languages were developed. One of us specified the first version of FORML to capture ORM constraints in textual form, and in the 1990s provided the patterns used to automatically generate verbalizations of constraints in some early ORM tools (InfoDesigner, InfoModeler, VisioModeler, Microsoft Visio for Enterprise Architects). At that stage, FORML was mainly an output language (while fact types could be entered in FORML, constraints and derivation rules could not be entered textually, but were instead verbalized from models that had been entered diagrammatically). The Language for Information Structure and Access

Descriptions (LISA-D), based on PSM, was specified [25] by researchers at Radboud University, and later extended to the Fact Calculus [26], but was not fully implemented. ConQuer, an ORM-based language for conceptual queries, was implemented in the ActiveQuery tool, which generated SQL code from ConQuer queries [5]. However, ActiveQuery did not allow models to be entered in textual form, and is no longer available. The Constellation Query Language (CQL) is currently under development in the Active Facts tool [23] to provide ConQuer-like functionality (a BNF grammar for CQL is accessible at [24]).

Outside the fact-oriented modeling community, several high level textual modeling and/or query languages were developed. Of those based on information modeling approaches, the most widely adopted is OCL, a formal language used to augment UML class models with rules that cannot be expressed graphically in UML [30, 37]. While useful, OCL has three main drawbacks: its attribute-based nature leads to semantic instability; its rule contexts are restricted to classes; and OCL expressions are often too technical for business users to understand and hence validate.

For example, consider the ORM schema in Fig. 1(a). The entity types Person and Gender are depicted as named, rounded, solid rectangles with their reference modes in parenthesis. The value type PersonTitle is depicted with a dashed line. Relationships are depicted as named sequences of role boxes (a role is a part in a relationship). Solid dots depict mandatory role constraints, bars depict uniqueness constraints, the value constraint on gender code is listed in braces, and the circled subset operator with connectors depicts a join-subset constraint. The explicit mandatory, uniqueness, and value constraints verbalize in FORML thus: **Each** Person is of **exactly one** Gender; **Each** Person has **exactly one** PersonTitle; **Each** PersonTitle is restricted to **at most one** Gender; **The possible values of** Gender **are** 'M', 'F'. Some person titles (e.g. 'Mr', 'Mrs', 'Ms', 'Lady') are restricted to a single gender, while others are not (e.g. 'Dr', 'Prof.'). The fact type PersonTitle is restricted to Gender is used to record any such restrictions.

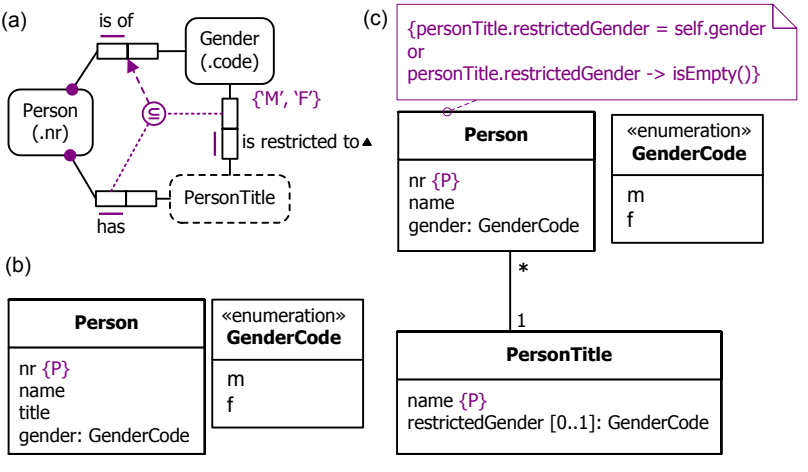


Fig. 1. (a) ORM schema captured (b) partly and (c) fully in UML

The join subset constraint ensures that the set of (Person, Gender) pairs projected from the person and gender roles in the path from Person through PersonTitle to Gender must be a subset of the (Person, Gender) pairs projected from the Person is of Gender fact type. The path at the subset end involves a conceptual join on PersonTitle. This constraint may be verbalized in FORML thus:

If a Person has **a** PersonTitle **that** is restricted to **some** Gender
then that Person is of **that** Gender.

Fig. 1(b) shows how a novice modeler might model this example in UML (although we've extended the UML notation here to include a {P} constraint for the primary value-based identifier). This loses the ability to express the subset constraint, because title is modeled as an attribute, so it cannot participate in an association. Attribute-based approaches are inherently unstable, because if ever one later needs to have an attribute play a role, the attribute needs to be remodeled, along with any rules or queries involving the attribute. Fig. 1 (c) shows how an experienced UML modeler might model this example. UML has no graphic notation for join subset constraints, so the subset constraint is instead captured by the OCL rule shown. This syntax is opaque to non-technical business people so is useless for validating the rule. While the intelligibility of OCL could in principle be addressed by a friendlier surface syntax, this has yet to occur, and would not solve the semantic instability problem.

Some textual languages for ER have been proposed (e.g. see section 16.3 of [22]), but these are limited in scope, and share with OCL the problem of semantic instability caused by an underlying attribute-based model.

Some business rules languages simply capture rules in a semi-natural language using patterns, while lacking the formal underpinnings to generate code (e.g. RuleSpeak [33]). Controlled natural languages are often linguistics-based, employing a formal, executable subset of natural language (typically English). Some of these languages use a linguistic, artificial intelligence approach, perhaps drawing upon existing public lexicons. Attempto Controlled English (ACE) [1, 35] supports a wide range of natural statements and queries, relying on interpretation rules (e.g. **and** has priority over **or**) to enable its text to be automatically and unambiguously translated into discourse representation structures, a syntactic variant of first-order logic. John Sowa's Common Logic Controlled English (CLCE) [36] has the full expressibility of first-order logic (FOL), while providing a semi-natural syntax that can be automatically translated into FOL. As discussed later, CLCE's use of untyped variables tends to make its expressions look more mathematical than natural.

In contrast to some controlled natural languages, Processable ENGlish (PENG) [35] uses a controlled lexicon of predefined function words as well as domain-specific content words that can be defined by the author on the fly. PENG texts can be deterministically translated into discourse representation structures or FOL for theorem proving. Like PENG, FORML restricts its lexicon to the model currently defined by the user. FORML derivation rules, constraints, and queries are constrained to the ORM model of interest (e.g. a derivation rule body for a new fact type must not refer to object types or fact types that are not already declared in the ORM schema).

Further details on controlled natural languages may be found on Rolf Schwitter's Website [34], as well as Jonathan Pool's review of controlled languages [32], which also includes an extensive list of references.

3 Overview of NORMA and Its FORML 2 Support

NORMA is an ORM 2 tool under development that is implemented as a plug-in to Microsoft Visual Studio. Most of NORMA is open-source, and a public domain version is freely downloadable [29]. Fig. 2 summarizes the main functions of the tool. Users may enter object types, fact types, and reference modes textually in FORML using the Fact Editor. These model components are stored in the conceptual model and automatically displayed in diagram form in the ORM diagrammer as well as in an explorer-layout in the Model Browser. Sample object and fact instances are entered in tabular format in the sample Population Editor.

At the time of writing, the public domain version requires ORM constraints and derivation rules to be entered in the ORM diagrammer or the Properties Window. Our modeling team at Logicblox recently extended the Model Browser to enable derivation rules for both fact types and subtypes to be formally captured and stored in a rules component of the conceptual model based on the role calculus [9].

Using mappers, ORM schemas may be auto-transformed into various implementation targets, including relational schemas for popular DBMSs (SQL Server, Oracle, DB2, MySQL, etc.), datalog, .NET languages (C#, VB etc.), and XML schemas. A Relational View extension displays the relational schemas in diagram form.

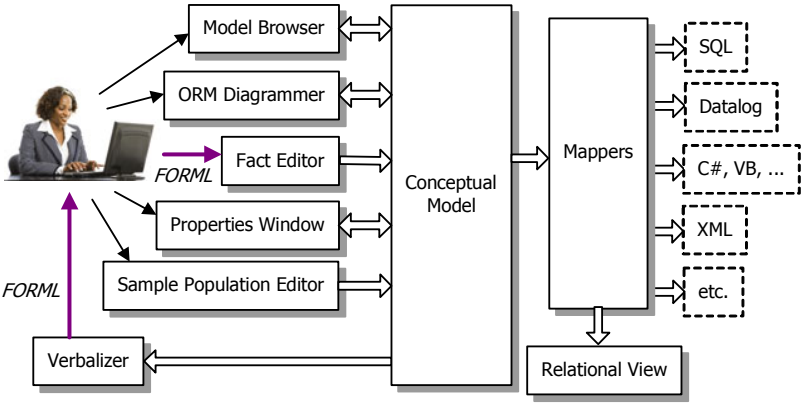


Fig. 2. Overview of main aspects of the NORMA tool

To facilitate validation of ORM models with domain experts, and to provide feedback to modelers on the meaning of ORM diagrams, the Verbalizer automatically verbalizes the models (or any selected part of them) in FORML. This makes use of FORML as an *output language*. For example, Fig. 3 is a screenshot from NORMA showing two fact types with spanning uniqueness constraints and a pair-exclusion constraint. For this screenshot, the top uniqueness constraint and the exclusion constraint have been selected. The rather verbose verbalization of the uniqueness constraint clarifies the *m:n* and set-based nature of the review fact type, and the verbalization of the exclusion constraint is also easily understood. Domain experts can validate the verbalization without needing to understand or even view the diagrams.

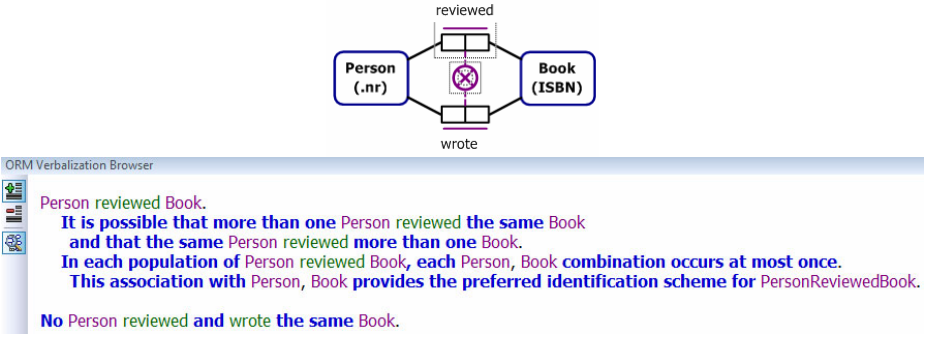


Fig. 3. NORMA screenshot showing verbalization of two selected constraints

Verbose *output* verbalizations are acceptable, since they are automatically generated, and are intended to clarify the semantics for nontechnical users. Indeed, in practice, industrial practitioners have typically rated the automated verbalization support in NORMA to be one of its most valuable features. However, for *inputting* FORML expressions, more concise formulations are often desirable. For example, the uniqueness constraint verbalized in Fig. 3 may be declared simply by appending "m:n" (for many-to-many) to the fact type entry.

ORM's rich expressivity makes full verbalization of ORM models a non-trivial task. One challenging aspect is verbalization of constraints or derivation rules involving join paths, of which the subset constraint in Fig. 1(a) provides a simple example. Some basic patterns for join constraint verbalization were considered in [18], and the role calculus framework for capturing derivation rules was introduced in [8]. The professional version of NORMA has just been extended to support verbalization of join constraints and of derivation rules entered in the Model Browser.

Previously, use of FORML as an *input language* has been restricted to entry of object types, fact types, and reference modes. It has been a long term goal for NORMA to enable users to enter full ORM models (including constraints and derivation rules) in purely textual form using the FORML language. As two initial but significant steps to meet this goal, in the last several months we have refined the FORML grammar, and implemented a prototype to enable most FORML constraints and derivation rules to be entered in an extended form of the Fact Editor.

An even longer term goal for FORML is to support its use as a *conceptual query language*. We have designed FORML to include query capability, and the work we have done on support of derivation rules can be leveraged to provide this capability (a query may be viewed as a request to derive information using asserted or derived fact types). However, in this paper our discussion of FORML focuses on its use for expressing *constraints and derivation rules*. The next section discusses the main features of FORML as well as various design guidelines used in its specification. Implementation aspects are discussed in Section 5.

4 FORML 2 Features and Design Guidelines

Formally, FORML 2 is based on sorted, first order logic plus bag and set comprehension, as well as making use of basic modal operators. Apart from a list of predefined functions, FORML predicates and types are restricted to those that have been declared in the ORM schema at the time the constraint or derivation rule is specified. By default, the *modality* of constraints is assumed to be alethic, although alethic modality may be explicitly included by prepending "**It is necessary that**" to constraints in positive form (e.g. **It is necessary that each** Person was born on **at most one** Date) or inserting "**It is impossible that**" to constraints in negative form (e.g. **For each** Person, **it is impossible that that** Person was born on **more than one** Date). For deontic counterparts, "**necessary**" and "**impossible**" is replaced by "**obligatory**" and "**forbidden**" respectively.

In FORML, object type names start with a capital letter to distinguish them from predicate text, and formal words are highlighted (e.g. by bolding, coloring, or use of special delimiters). Fact types may be objectified, and then treated as object types. For example: Patient is allergic to Drug (**objectify as** Allergy); Allergy was detected on Date. *Hyphen binding* (forward or reverse) may be used to bind modifiers to terms, ensuring quantifiers are placed for natural verbalization. For example, "most- influential Person" and "Person of highest -influence" keep the modifiers with Person irrespective of quantifiers.

FORML expressions typically read more naturally than expressions in other fact-oriented languages. For example, the LISA-D based fact calculus rule "NO Official-paper of A Car being returned BUT NOT being returned" [24] is verbalized in FORML as "**No** Car **that** is returned has **some** OfficialPaper **that** is **not** returned".

FORML allows use of *pronouns* such as "**that**" (for impersonal types) and "**who**" (for personal types) as well as *typed variables, possibly subscripted* (e.g. Person_1) instead of untyped variables (e.g. x, y, z_1, z_2) for *correlation*. Compare the CLCE rule

> If some person x is a parent of some person y, and the person y is a parent of some person z, then the person x is a grandparent of the person z.

with the FORML rule, rendered in *relational style* (which uses predicate readings)

> Person_1 is a grandparent of Person_2 **iff**
> Person_1 is a parent of **some** Person_3 **who** is a parent of Person_2.

Here, "**iff**" may be expanded to "**if and only if**". Variables in the head of a rule are assumed to be universally quantified, but such quantifications may be made explicit by prepending a **for-each** clause, e.g.

> **For each** Person_1 **and** Person_2, Person_1 is a grandparent of Person_2 **if and only if**
> Person_1 is a parent of **some** Person_3 **who** is a parent of Person_2.

FORML rules may also be rendered in *attribute style* (which uses role names) using either dot notation or of-notation (which reverse the dot order). For example, assuming the role name "parent" is declared, the above rule may be stated in attribute style as either of the following: **For each** Person, grandparent = parent.parent.; **For each** Person, grandparent = parent **of** parent.

Attribute style typically gives more compact expression for arithmetic derivation rules, e.g. **For each** Invoice, total = **sum**(lineItem.(quantity * unitprice). *Mixed style* (which allows a combination of relational and attribute styles) is also permitted.

Conjunction and *disjunction* is handled using "**and**" and "**or**" (treated as inclusive-or). Like English (but not ACE), FORML gives these operators equal precedence, so commas or brackets are used where necessary to disambiguate combinations. *Negations* may be expressed by prepending "**It is false that**" but may often be rendered more naturally using "**no**", e.g. **Each** NonDriver **is a** Person **who** drives **no** Car.

Currently, FORML rules assume that all types and predicates are predeclared, and no linguistic machinery is employed to recognize variations in number (e.g. plurals) or voice etc. Hence all object type names should be singular (e.g. Person, not Persons or People). *Pluralization* in rules is obtained by using "**instances of**". For example, instead of **Each** Person is a child of **at most** 2 Persons, we instead say **Each** Person is a child of **at most** 2 **instances of** Person. Moreover, NORMA is incapable of deducing that the fact type reading "Person drives Car" is equivalent to "Person does drive Car" (thus eliminating generation of alternate negations such as **Each** NonDriver **is a** Person **who** does **not** drive **any** Car.). However, users are free to manually provide multiple readings for the same fact type, any of which may then be used in rules.

FORML rules may navigate freely across ORM schemas, introducing joins, operators, and functions, so care is needed to ensure disambiguation. For example, Fig. 3 includes three equivalent derivation rules for the subtype RecentlyLicensedDrivingDoctor. Rule (1), in relational style, uses instance-level "**is**" predicates to navigate to supertypes. The other rules in attribute style need to disambiguate reference to the relevant licenseDate role. Rule (2) uses an "**as**" clause to *cast* DrivingDoctor as Doctor, then accesses the closest licenseDate role. Rule (3) is similar, but uses the implicit rolename "asDoctor" for the casting.

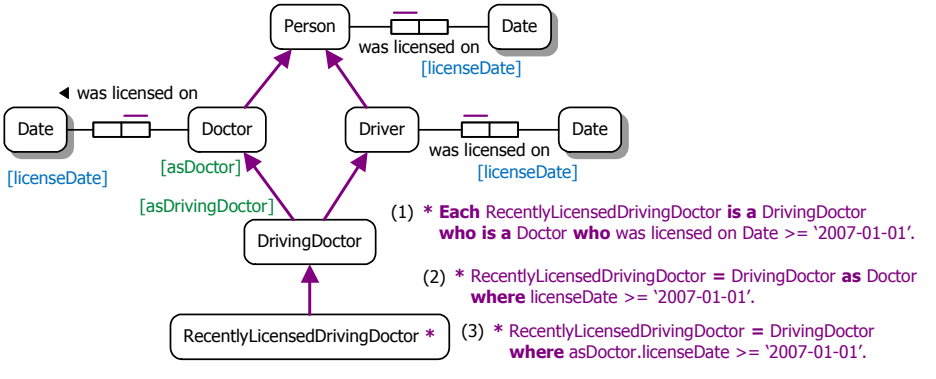


Fig. 4. Three equivalent subtype definitions with unambiguous reference to licenseDate

This illustrates the following procedure. For a given subtype, if multiple far roles have the same name, then disambiguate as follows:

If a direct far role of the subtype has that role name
then that role is chosen
else if only one of its supertype chains includes a supertype with a far role with that name
    then choose the first such supertype far role found (moving up the chain)
    else explicitly include the relevant subtyping connections before the rolename
        (use dot notation or prepend "**as** " to the name of each sub/supertype being navigated to).

While FORML permits all type variables to be subscripted (as in ConQuer [5]), it also includes various procedures to minimize the use of subscripts, thus allowing more natural formulation. For example, object variables introduced in the body of a fact type derivation rule are assumed to be existentially quantified. However, "**some**" may be prepended to make the quantification explicit, and "**that**" must be prepended when correlating unsubscripted variables to previous, unsubscripted occurrences of that variable. For example, the following two explicit formulations are allowed, but may be abbreviated to the later formulations:

*Person drives imported- Car **iff** Person drives Car **that** is imported from **some** Country.
*Person$_1$ is a father **iff** Person$_1$ is a parent of **some** Person$_2$ **and** Person$_1$ is male.

*Person drives imported- Car **iff** Person drives Car **that** is imported from Country.
*Person$_1$ is a father **iff** Person$_1$ is a parent of Person$_2$ **and** Person$_1$ is male.

There is no space here to cover all of FORML's disambiguation rules, but the above examples are representative of the kinds of rules adopted.

5 FORML 2 Grammar and Implementation

After specifying a basic input grammar for FORML 2 in Extended Backus-Naur Form (EBNF) [23], supplemented by some metarules (e.g. fact type readings referenced in rules must be pre-declared), we implemented a prototype of the grammar as an extension to NORMA's Fact Editor. To test the grammar, we used the ANTLR parser generator [30] and the ANTLRWorks development environment [4]. ANTLRWorks accepts EBNF-grammars and also supports a range of non-context-free parsing aids (e.g. gated semantic predicates). Our main reasons for choosing ANTLR include good tool support, code generation to C#, LL(*) "infinite" lookahead, and non-EBNF features like semantic predicates. These features facilitated quick iterations of the test/implement cycle [20]. A test suite of sample rules was constructed in collaboration with some ORM modelers. The test suite and a sample parse tree is accessible at http://home.kpn.nl/wijbe113/.

A FORML sentence is a sequence of textual items such as object type names, full or partial predicate readings, role names, *formal items* (operators, quantifiers, pronouns, etc.), constants (e.g. individual names or numbers), and punctuation marks (e.g., ",", "."). Object type terms start with a capital letter. Subscripts distinguish object variables of the same type. Formal items are comprised of *pseudo-reserved words*, and are displayed in a different *text style* (e.g. **bold**). Words within formal items are not fully reserved, since some of them may be used in a predicate reading. For example, in the following derivation rule, the third "a" is a formal item, but not the other instances of "a". While informally all four instances of "a" have the meaning of an existential quantifier, only the bolded "a" is formally interpreted as such.

Person$_1$ is a grandfather of Person$_2$ **iff** Person$_1$ is a parent of **a** Person$_3$ **who** is a parent of Person$_2$.

We refer to this syntax as *front-end FORML* since we plan to support it in the user interface (an extended Fact Editor) for inputting FORML text. However, while the verbalizer fully supports rich text controls for output FORML, to expedite the implementation of the prototype for input FORML, we delayed rich text editing support for that, instead using a *back-end FORML* grammar that distinguishes formal items and

subscripts by addition of special characters. For example, formal items are included in braces, and subscripts are indicated by prepending dollar signs. The above rule appears in back-end FORML thus: Person$1 is a grandfather of Person$2 {iff} Person$1 is a parent of {a} Person$3 {who} is a parent of Person$2{.}. For convenience, the other examples in this paper are displayed in front-end FORML.

Now consider the following derivation rule. The first occurrence of "**that**" appears just after an object term (Office), so is a relative pronoun used to perform a conceptual join on Office. The second occurrence of "**that**" precedes an object term (Building) so is an indicative pronoun referencing an earlier occurrence of that term. In back-end FORML these occurrences are distinguished as {that} and {jointhat}.

Employee works in Building **iff** Employee works in **some** Office **that** is in **that** Building.

The body of this rule (the part after "**iff**") contains two parts: Employee works in **some** Office; and **that** is in **that** Building. The second part starts with a join pronoun that stands for the referenced object variable. Each of these parts is called a Quantified Correlated Reading Instance (QCRI). Every QCRI contains a reading of exactly one fact type. This reading may have any arity (1 or more). It may have front text preceding the first object term, end text after the last object term, as well as hyphen-bound text. Each object type term may be subscripted, and may be preceded by quantifiers (e.g. "**some**", "**each**", "**no**") or by the relative pronoun "**that**".

As a simple example of an attribute-style rule, consider the following derivation rule, which navigates using far-role names.

For each Window, area = height * width.

In back-end FORML, this is written {For-each} Window, [area] = [height] * [width] {.}. Fig. 5 shows the parse tree for this input, as generated by the ANTLRWorks interpreter.

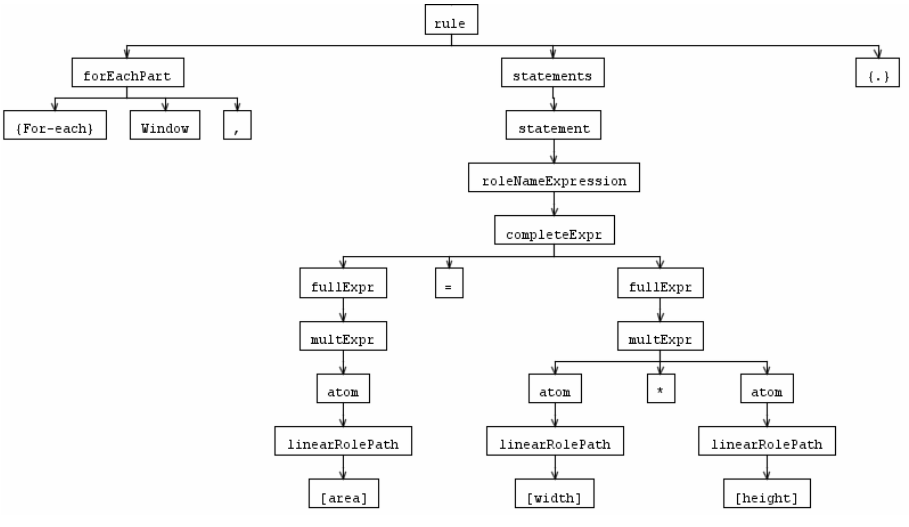


Fig. 5. Parse tree generated by ANTLRWorks for the area derivation rule

Another nice feature of the ANTLRWorks tool is its ability to automatically display the EBNF as railroad diagrams. Fig. 6 shows an example from our grammar.

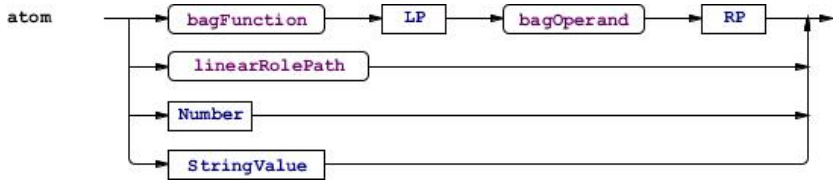


Fig. 6. Railroad diagram for EBNF definition of atom

Validating back-end FORML requires more than simple syntactic checks based on the EBNF grammar. For example, fact type readings referenced in rules must already be present in the ORM schema. When some text is parsed as a QCRI, its fact type reading is extracted and compared to the fact type base. The extraction normalizes whitespace, deletes subscripts and quantifiers, and if the QCRI is contracted then the last object type name from the previous QCRI is prepended to the reading. The following ANTLR rule matches a contractedQCRI.

```
contractedQCRI
@after{
$linearPath::lastOT = LastOTFrom($linearPath::lastOT,$pt);
}
        :       pt+=LCWord+
                (
                (quantifier ( pt+=LCWord+)?)? pt+=ObjectType Sub?
                (pt+=LCWord+ (quantifier ( pt+=LCWord+ )? )? pt+=ObjectType)*
                pt+=LCWord*)?
                // validating semantic predicate
                {IsReading($linearPath::lastOT,$pt)}?;
```

The two sections in braces contain the semantic actions. The last encountered object type is stored as a variable LastOT that is defined in a higher scope. The method calls LastOTFrom() and IsReading() call methods that are located in a manually coded extension of the generated parser. C# partial classes provide an elegant solution for extending the generated parser with method implementations.

Now consider the subtype derivation rule just below. Initially, we treated this as ambiguous, with the two different meanings shown in the subsequent rules. Detecting such cases requires checking whether a contracted QCRI is preceded by end text.

***Each** LazyDogOwner **is a** Person **who** owns **some** Dog **that** barks **and** is lazy.
***Each** LazyDogOwner **is a** Person **who** owns **some** Dog **that** barks **where that** Dog is lazy.
***Each** LazyDogOwner **is a** Person **who** owns **some** Dog **that** barks **where that** Person is lazy.

We are now considering allowing such expressions for input, by adopting this implicit previous subject rule: If a clause beginning with "**and**" or "**or**" immediately precedes predicate occurrence *R2*, and the previous predicate occurrence *R1* has no front text and is either a unary or an infix binary, then the subject of *R2* is the subject of *R1*.

For output verbalization, however, the expanded form (second formulation above) would be used to ensure users are aware of the interpretation taken.

6 Conclusion

This paper discussed the use of FORML 2 as a high level textual language that can be used with the NORMA tool for both input of ORM 2 models and output verbalization of ORM 2 models, including constraints and derivation rules. The combination of automatic transformation between textual and diagrammatic forms, and the wide range of rules covered, distinguishes our approach from many other approaches based on controlled natural languages. While the work on output FORML is relatively mature, the work on input FORML is still in its early stages, and further research is needed to exploit the full potential of this approach.

Future plans include a rich text editor for inputting front-end FORML rather than back-end FORML. The translation of derivation rules in input FORML to the role calculus form also requires more research. Use of a GLR parser generator could be considered as an alternative to the current LL(*) grammar. Instead of asking the user to disambiguate right away, a GLR parser tolerates ambiguity and generates a parse forest from which the intended parse tree can be chosen. Other plans include support for pluralization, dynamic rules [3], conceptual queries, and non-English languages.

References

1. Attempto project (Attempto Controlled English), http://attempto.ifi.uzh.ch/site/
2. Bakema, G., Zwart, J., van der Lek, H.: Fully Communication Oriented Information Modelling. Ten Hagen Stam (2000)
3. Balsters, H., Halpin, T.: Formal Semantics of Dynamic Rules in ORM. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2008. LNCS, vol. 5333, pp. 699–708. Springer, Heidelberg (2008)
4. Bovet, J., Parr, T.: ANTLRWorks: an ANTLR grammar development environment. In: Software: Practice and Experience, vol. 38(12), pp. 1305–1322. John Wiley, Chichester (2008)
5. Bloesch, A., Halpin, T.: Conceptual queries using ConQuer-II. In: Embley, D.W. (ed.) ER 1997. LNCS, vol. 1331, pp. 113–126. Springer, Heidelberg (1997)
6. Business Rules Solutions Website on RuleSpeak, http://www.rulespeak.com/en/
7. Chen, P.P.: 'The entity-relationship model—towards a unified view of data'. ACM Transactions on Database Systems 1(1), 9–36 (1976)
8. Curland, M., Halpin, T., Stirewalt, K.: A Role Calculus for ORM. In: Meersman, R., Herrero, P., Dillon, T. (eds.) OTM 2009 Workshops. LNCS, vol. 5872, pp. 692–703. Springer, Heidelberg (2009)
9. Halpin, T.: ORM 2. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2005. LNCS, vol. 3762, pp. 676–687. Springer, Heidelberg (2005)
10. Halpin, T.: ORM/NIAM Object-Role Modeling. In: Bernus, P., Mertins, K., Schmidt, G. (eds.) Handbook on Information Systems Architectures, 2nd edn., pp. 81–103. Springer, Heidelberg (2006)

11. Halpin, T.: Modality of Business Rules. In: Siau, K. (ed.) Research Issues in Systems Analysis and Design, Databases and Software Development, pp. 206–226. IGI Publishing, Hershey (2007)
12. Halpin, T.: Fact-Oriented Modeling: Past, Present and Future. In: Krogstie, J., Opdahl, A., Brinkkemper, S. (eds.) Conceptual Modelling in Information Systems Engineering, pp. 19–38. Springer, Berlin (2007)
13. Halpin, T.: A Comparison of Data Modeling in UML and ORM. In: Khosrow-Pour, M. (ed.) Encyclopedia of Information Science and Technology, 2nd edn., Information Science Reference, Hershey PA, USA, vol. II, pp. 613–618 (2008)
14. Halpin, T.: Object-Role Modeling. In: Liu, L., Tamer Ozsu, M. (eds.) Encyclopedia of Database Systems. Springer, Berlin (2009)
15. Halpin, T.: Predicate Reference and Navigation in ORM. In: Meersman, R., Herrero, P., Dillon, T. (eds.) OTM 2009 Workshops. LNCS, vol. 5872, pp. 723–734. Springer, Heidelberg (2009)
16. Halpin, T.: Object-Role Modeling: Principles and Benefits. International Journal of Information Systems Modeling and Design 1(1), 32–54 (2010)
17. Halpin, T., Morgan, T.: Information Modeling and Relational Databases, 2nd edn. Morgan Kaufmann, San Francisco (2008)
18. Heath, C.: The Constellation Query Language. In: Meersman, R., Herrero, P., Dillon, T. (eds.) OTM 2009 Workshops. LNCS, vol. 5872, pp. 682–691. Springer, Heidelberg (2009)
19. Heath, C.: ActiveFacts Website (2009), http://dataconstellation.com/ActiveFacts/
20. Hevner, A., March, S., Park, J., Ram, S.: Design Science in Information Systems Research. MIS Quarterly 28(1), 75–105 (2004)
21. ter Hofstede, A., Proper, H., van der Weide, T.: Formal definition of a conceptual language for the description and manipulation of information models. Information Systems 18(7), 489–523 (1993)
22. Hoppenbrouwers, S., proper, H., van der Weide, T.: Fact Calculus: Using ORM and Lisa-D to Reason about Domains. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2005. LNCS, vol. 3762, pp. 720–729. Springer, Heidelberg (2005)
23. ISO 1996, Information technology – Syntactic metalanguage – Extended BNF, ISO/IEC 14977 (1966), http://www.iso.org/iso/catalogue_detail.htm?csnumber=26153
24. Meersman, R.: The RIDL Conceptual Language. Int. Centre for Information Analysis Services, Control Data Belgium, Brussels (1982)
25. Nijssen, M., Lemmens, I.: Verbalization for Business rules and Two Flavors of Verbalization for Fact Examples. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2008. LNCS, vol. 5333, pp. 760–769. Springer, Heidelberg (2008)
26. NORMA (Natural ORM Architect) tool download site for public-domain version, http://www.ormfoundation.org/files/folders/norma_the_software/default.aspx
27. Object Management Group 2003, UML 2.0 Superstructure Specification, http://www.omg.org/uml
28. Object Management Group 2005, UML OCL 2.0 Specification, http://www.omg.org/docs/ptc/05-06-06.pdf
29. Object Management Group 2008, Semantics of Business Vocabulary and Business Rules (SBVR), http://www.omg.org/spec/SBVR/1.0/
30. Parr, T.: The Definitive ANTLR Reference: Building Domain-Specific Langauges, 1st edn. Pragmatic Bookshelf, Raleigh (2007)

31. Pool, J.: Can Controlled Languages Scale to the Web? In: Proc. CLAW 2006: 5th International Workshop on Controlled Language Applications (2006), http://utilika.org/pubs/etc/ambigcl/clweb.html
32. Schwitter, R.: Controlled Natural Languages, http://sites.google.com/site/controllednaturallanguage/
33. Schwitter, R.: PENG (Processable English) (2007), http://web.science.mq.edu.au/~rolfs/peng/
34. Sowa, J.: Controlled English (2004), http://www.jfsowa.com/logic/ace.htm
35. Sowa, J.: Common Logic Controlled English (2004), http://www.jfsowa.com/clce/specs.htm
36. Warmer, J., Kleppe, A.: The Object Constraint Language, 2nd edn. Addison-Wesley, Reading (2003)

Specifying Structural Properties and Their Constraints Formally, Visually and Modularly Using VCL

Nuno Amálio, Pierre Kelsen, and Qin Ma

University of Luxembourg, 6, r. Coudenhove-Kalergi, L-1359 Luxembourg
{nuno.amalio,pierre.kelsen,qin.ma}@uni.lu

Abstract. The value of visual representations in software engineering is widely recognised. This paper addresses the problem of formality and rigour in visual-based descriptions of software systems. It proposes a new language, VCL, designed to be visual, formal and modular, targeting abstract specification at level of requirements, and that aims at expressing visually what is not visually expressible using mainstream visual languages, such as UML. This paper presents and illustrates VCL's approach to structural modelling based on the VCL notations of structural and constraint diagrams with a case study. VCL's contributions lie in its modularity mechanisms, and the support for two alternative styles of visual constraint modelling (one closer to set theory expressions and based on Euler diagrams, the other closer to predicate calculus and based on object graphs).

Keywords: formal modelling, visual languages, Z.

1 Introduction

The value of visual representations for problem solving is widely recognised [1]. In software engineering, visual languages have been advocated for decades [2]; this importance is demonstrated in practice: visual formalisms, such as UML, are the choice when it comes to software systems modelling [3,4].

The visual formalisms that most software engineers use, such as UML, are known as semi-formal methods [5,6]; semi-formal because they were designed to have a formal syntax, but no formal semantics. Although there have been successful formalisations of semantics for such languages (e.g subsets of UML, see [5]), they are mostly used without a formal semantics. The lack of formal semantics brings numerous problems [7]: (a) it is difficult to be precise and have a good sense of what is being specified, (b) models are prone to ambiguities and inconsistencies and (c) it is not possible to semantically analyse models mechanically. Another problem is that they cannot express diagrammatically a large number of properties of software systems; this is why UML is accompanied by the textual Object Constraint Language (OCL).

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 261–273, 2010.

Formal methods, such as Z [8] and Alloy [9], embody semantically sound languages. They do not suffer from the semantic-related problems of their semi-formal counterparts. However, despite some success stories, formal methods have not been taken up by practitioners [10,5], being used only in the safety-critical niche [10]. Like others [2,11], we see in the visual and formal a promising combination to enhance the practicality and adoptability of formal techniques.

This paper presents the Visual Contract Language (VCL) [12,13]. VCL is designed for abstract specification of software systems visually, formally and modularly. Visually because visual representations favour human-processing. Formally because formality enhances precision and enables mechanical semantical analysis. Modularly because modularity helps tackling problem complexity by enabling problem decomposition.

This paper presents design of VCL for structural modelling with a case study. This is based on VCL notations of *structural* and *constraint* diagrams. The paper is as follows:

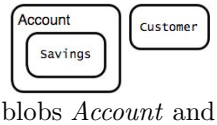
- It presents design of VCL [12,13], highlighting VCL's modularity mechanisms and its support for two alternative styles of constraint specification (one is set-theoretic and the other is akin to predicate-calculus).
- It illustrates formal semantics outline that accompanies VCL's design by giving examples of how VCL diagrams would be represented formally.
- It presents some initial results towards development of tool support for VCL[1].
- It shows how invariants, usually described textually in a formal language (such as OCL) in UML-based models, can be described visually using VCL.

2 Overview of Structural VCL

VCL has been designed to have a minimal set of visual primitives. Because these primitives are used in different types of diagrams and in different contexts, they have a core meaning that varies slightly with the context. In VCL presented here, same visual concept can be used in both structural and constraint diagrams.

The abstract syntax of VCL notations of structural and constraint diagrams presented here is given in [14]; it is defined formally using OO *metamodels* specified in the Alloy modelling language [9].

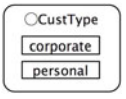
2.1 Visual Primitives



VCL's *blob* concept is like an Euler circle: a rounded contour denoting a set. Topological notions of *enclosure* and *exclusion* represent subset and disjoint relations. To the left, blobs *Account* and *Customer* represent disjoint sets of objects; *Savings* is a subset of *Account*.

[1] The *visual contract builder tool*, `http://vcl.gforge.uni.lu`

VCL's concept of an *object* or *atom* is represented as a rectangle. Objects denote an element of some set. To the left, *MrSmit* is an object of blob *Customer*.



Blobs may enclose objects as well as other blobs, and they may be defined in terms of the things they enclose by preceding the blob's label with symbol ◯. To the left, *CustType* is defined in this way by enumerating its elements.

Edges connect both blobs and objects to define various kinds of relations. There are two kinds of edges: property and relational.

Property edges, represented visually as directed arrows labelled with a name, denote or refer to some property possessed by all elements of the set (e.g. *balance* to the left); they are like *class attributes* in the object-oriented (OO) paradigm.

Relational edges are represented as labelled directed lines, where direction is indicated by arrow symbol above the line. Their label is within a blob because they denote a set of tuples and may be placed inside blobs. Relational edges define or refer to some conceptual relation between blobs (*associations* in OO)[2] (e.g. *Holds* to the left).

To indicate that some model structure(s) are subject to constraints, VCL uses *constraints* (e.g. *TotalBalIsPositive* to the left), which are labelled with the constraint name they refer to.

2.2 Structural Diagrams

Structural diagrams (SDs) define the structures that make the system's state space. They describe main problem domain concepts as blobs, their internal state as property edges, their conceptual relations as relational edges, and invariants as constraint references (see Fig. 1 for an example).

In SDs, there are two types of blobs: *domain* and *value*. Domain blobs, represented using a bold line, are part of the state of overall system; they are dynamic and need to be maintained. Value blobs define an immutable set of values that do not need to be maintained. In Fig. 1, *Account* and *Customer* are domain blobs; *Name* is a value blob. In SDs, blobs may be defined by enumerating its constituent objects; blobs *CustType* and *AccType* are defined in this way.

2.3 Constraint Diagrams

Constraint diagrams (CntDs) are made of three compartments: *name*, *declarations* and *predicate* (see Fig. 2 for an example). The declarations compartment introduces variable names together with other constraints being imported. The predicate compartment actually defines the constraint. A predicate can be formed of objects, blobs, relational and property edges as in CntD *AccSavings*

[2] Relational edges denote a relation between sets or a tuple depending on whether they connect blobs or objects.

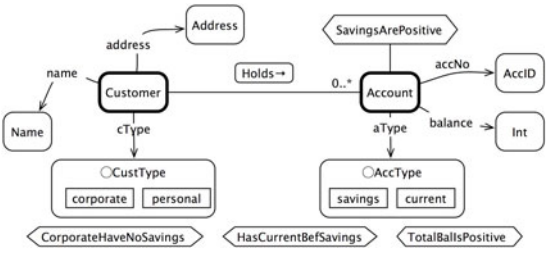


Fig. 1. Structural diagram of simple Bank

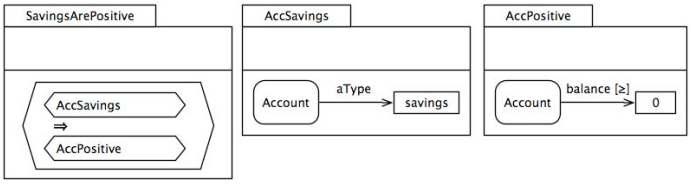


Fig. 2. Constraint diagrams for *Account* invariant *SavingsArePositive*

of Fig. 2 (p. 264), or made up of a constraint reference expression as in CntD *SavingsArePositive* of Fig. 2. VCL enables combination of constraint diagrams using logical operators, namely: negation, conjunction, disjunction, implication, and universal and existential quantification.

2.4 Semantics

VCL's design presented here is accompanied by an outline of its formal semantics. VCL embodies a *generative* (or translational) approach to semantics. It is to be used together with a textual formal specification language, *target language*, that sits in the background and a target language semantic model. The semantics of a VCL specification is the generated target language specification. This paper uses Z as target language and ZOO [15,5] as semantic domain. ZOO is an abstract OO semantic domain for language Z. We use this way our previous result, enabling us to focus on the visual aspects of VCL.

The VCL diagrams are mapped into a ZOO model, which comprises Z structures representing the various elements of a VCL model. Semantically, a blob is a set, property edges are properties shared by all objects of the set, relational edges are relations between sets, ensembles are collections of sets and relations, constraints are predicates that restrict some state structure or ensemble. Various VCL model elements are represented as Z schemas that can be combined in various ways.

3 Running Example

VCL is illustrated here with simple Bank case study, which is also used to illustrate the ZOO semantic domain in [15,5]. The case study's requirements are given in table 1.

The following presents VCL's structural and constraint diagrams describing *simple Bank*. The outline of VCL's Z semantics is illustrated here by presenting how VCL diagrams would be represented in ZOO. Full Z specification resulting from VCL semantics outlined here is available online [16].

Table 1. Requirements of the simple Bank system

R1	The system shall keep information of customers and their Bank accounts. A customer may hold many accounts, but an account is held by one customer.
R2	A customer shall have a *name*, an *address* and a *type* (either company or personal).
R3	A Bank account shall have an account number, a balance indicating how much money there is in it, and its type (either current or savings).
R4	Savings accounts cannot have negative balances.
R5	The total balance of all Bank's accounts must not be negative.
R6	Customers of type *corporate* cannot hold savings accounts.
R7	To open a savings account, customer must already hold a current account with the Bank.

4 Defining the Structures That Make the State Space

VCL SDs define state structures and identify the invariants that constrain them. There are two types of invariants. *Local invariants* are attached to some blob and they affect and are described in the scope of associated blob; they are known as class invariants in OO paradigm. *Global invariants* affect and are described in the scope of an ensemble of state structures as defined by some SD.

4.1 Simple Bank System

Figure 1 presents VCL SD of simple Bank. It is as follows:

- Domain blobs *Customer* and *Account* represent main problem domain concepts (requirement $R1$). Property edges *name*, *cType* and *address* define properties of *Customer* (Requirement $R2$); *accNo*, *balance* and *aType* define properties of *Account* (Requirement $R3$).
- Blobs *CustType* and *AccType* are defined by enumeration (symbol ◯). *CustType* has elements *corporate* and *personal*; *AccType* has elements *savings* and *current*.
- Relational edge *Holds* relates customers and their accounts. UML-style multiplicity constraints say that a customer may have many accounts and that an account is held by one *Customer* (Requirement $R1$).
- Several invariants constrain state of the system. *SavingsArePositive* is local. Remaining invariants are global: *CorporateHaveNoSavings* (Requirement $R6$), *HasCurrentBefSavings* (Requirement $R7$) and *TotalBalIsPositive* (Requirement $R5$).

4.2 Z Representation

Z representation of SDs follows ZOO approach for construction of state spaces outlined in [15,5]. Value blobs that are not enumerations are defined as given sets, enumerations as free types, domain blobs as promoted abstract data types (ADTs), and relational edges as Z relations. Finally, the ensemble of state structures defined by overall SD is built as a conjunction of those Z schemas representing domain blobs and relational edges. The following gives Z definitions of blobs *Name*, *Address*, *CustType* and *Customer*, relational edge *Holds* and state of ensemble for SD of Fig. 1.

$$[Name, Address]$$

$$CustType ::= corporate \mid personal$$

$$\begin{array}{|l} Customer \\ \hline name : Name \\ address : Address \\ cType : CustType \\ \hline \end{array} \qquad \begin{array}{|l} SCustomer \\ \hline sCustomer : \mathbb{O}\, Customer \\ stCustomer : (\mathbb{O}\, Customer) \pfun Customer \\ \hline \mathrm{dom}\, stCustomer = sCustomer \\ \hline \end{array}$$

$$\begin{array}{|l} AHolds \\ \hline Holds : \mathbb{O}\, Customer \rel \mathbb{O}\, Account \\ \hline \end{array} \qquad \begin{array}{|l} BankSt \\ \hline SCustomer;\ SAccount;\ AHolds \\ \hline \end{array}$$

5 Constraining the State Space

VCL's constraint diagrams enable specification of constraints in two styles. One is close to set theory and is based on blob constructions such as insideness and shading. The other is closer to predicate calculus and is based on object graphs.

As the examples given below show, CntDs are modules that can be composed in various ways.

5.1 Defining Constraints with Blobs

Blobs introduced in a SD are the building blocks of a VCL model. From them, derived blobs are defined for the purpose of constraining the state space. In addition to the blob relations of inclusion and exclusion, in CntDs blobs can be shaded to say that the denoted set must be empty.

Invariant *SavingsArePositive***.** This local invariant is described in Fig. 2 using three CntDs. All declarations compartments are empty because no extra declarations of names are required to describe the constraint. Invariant is described as follows:

- CntD *AccSavings* defines a predicate describing all those objects of *Account* whose property *aType* is equal to value *savings*.
- CntD *AccPositive* defines a predicate describing all those objects of *Account* whose *balance* must be greater or equal to 0 [3].

[3] In CntDs, property edges link some blob or object to some expression; by default they denote equality, unless other relational operator is explicitly provided. Above, *aType* edge denotes equality, but *balance* denotes $\geq$.

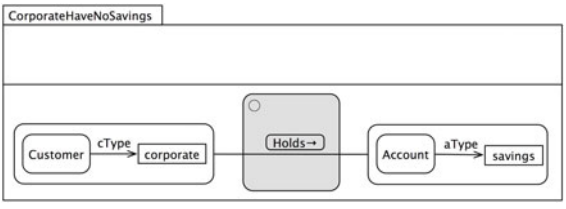


Fig. 3. Constraint diagram for invariant *CorporateHaveNoSavings*

– Finally, CntD *SavingsArePositive* defines actual constraint by saying CntD *AccSavings* implies *AccPositive*, which means that predicates encapsulated by these CntDs are being related using logical implication.

Z representation of this invariant comprises a Z schema for each CntD:

$$AccSavings \;==\; [\; Account \mid aType = savings \;]$$
$$AccPositive \;==\; [\; Account \mid balance \geq 0 \;]$$
$$SavingsArePositive \;==\; [\; Account \;\mid\; AccSavings \;\Rightarrow\; AccPositive \;]$$

Invariant *CorporateHaveNoSavings***.** This invariant is described in Fig. 3. Blob on the left restricts *Customer* to those objects whose property *cType* has value *corporate*. Blob on the right restricts *Account* to those objects whose property *aType* has value *savings*. Shadded blob in the middle captures relation *Holds* restricted to those tuples with corporate customers and savings accounts; shading says that set must be empty, giving required meaning.

Z representation of this invariant is as follows:

CorporateHaveNoSavings

$BankSt$

$$\{oC : sCustomer \mid (stCustomer\ oC).cType = corporate\} \lhd Holds \rhd \{oA : sAccount \mid (stAccount\ oA).aType = savings\} = \varnothing$$

Invariant *HasCurrentBefSavings***.** This invariant is described in Fig. 4 using three CntDs. CntD *CustsWithCurrentDef* defines set of customers with current accounts (*CustCurr*). CntD *CustsWithSavingsDef* defines set of customers with savings accounts (*CustSav*). Finally, CntD *HasCurrentBefSavings* says *CustSav* is subset of *CustCurr*; these names refer to same object in the different diagrams.

Constraint importing results in importing of names. When an imported name is not explicitly declared, then it is hidden. In *HasCurrentBefSavings* *CustSav* and *CustCurr* are not declared, so they are hidden. Note the use of *insideness* property of blobs to capture domain of relation *Holds*. Blobs *CustsSav* and *CustsCurr* are defined (symbol ◯) by having inside the blobs representing the relation and set that is in domain of relation; this means that we are capturing the domain of relation *Holds* subject to restrictions as defined in constraint. See [16] for Z definition of this invariant.

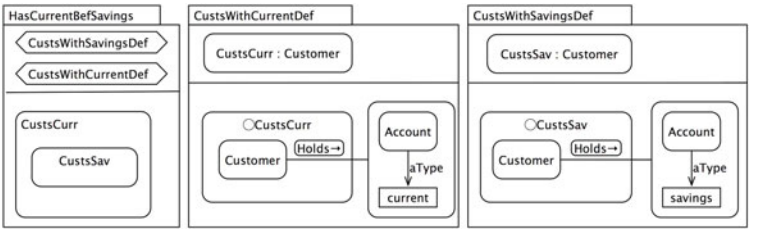


Fig. 4. Constraint diagram for invariant *HasCurrentBefSavings*

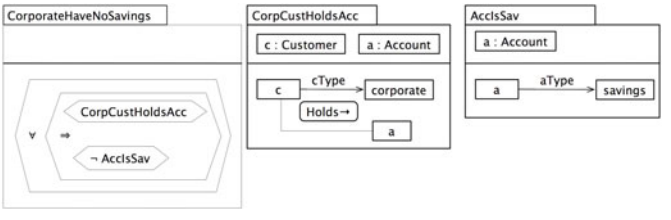


Fig. 5. Constraint diagram of *CorporateHaveNoSavings* using objects

5.2 Expressing Constraints with Objects and Quantifiers

Following CntDs illustrate use of objects and quantifiers to express constraints in a predicate-calculus style.

Invariant *CorporateHaveNoSavings*. Figure 5 gives an alternative formulation of this invariant to that given in Fig. 3, which is formulated using blobs. This defines CntDs *CorpCustHoldsAcc* (customer c is of type *corporate* and holds account a) and *AccIsSav* (account a is of type *savings*). CntD *CorporateHaveNoSavings* then says that the former implies the negation of the latter to say that a customer of type corporate must not have a savings account. Universal quantifier asserts that implication must hold for all customers c and accounts a. All variables are bound by the quantifier in both diagrams; name a in two different diagrams refers to same object. See [16] for Z definition of this invariant.

Invariant *HasCurrentBefSavings*. Figure 6 expresses this constraint by saying that all customers having a savings must also have a current account. A quantifier applied to a constraint binds all its variables, except when a variable has its scope extended by a *communication edge*. In Fig. 6, c's scope is extended in this way; hence, it is not bound by the two existential quantifiers. It is, however, bound by universal quantifier in *HasCurrentBefSavings*. See [16] for Z definition of this invariant.

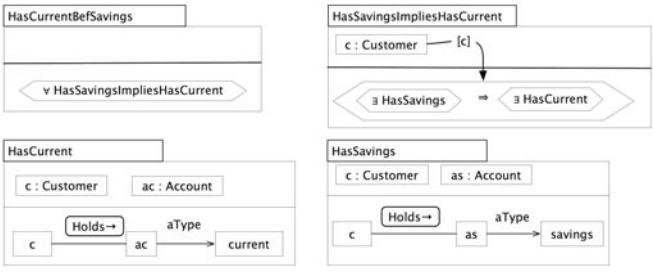


Fig. 6. Constraint diagram of *HasCurrentBefSavings* expressed using objects

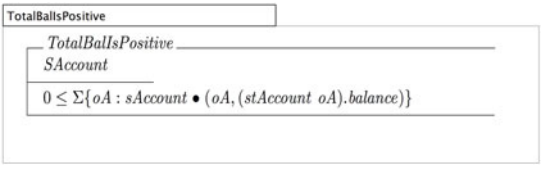


Fig. 7. Z constraint *TotalBalanceIsPositive* embedded in VCL constraint diagram

5.3 Constraints That Cannot Be Expressed Visually

Not all constraints can be expressed visually in VCL. *TotalBalanceIsPositive* of Bank is such a constraint. VCL gives the specifier the choice of writing a constraint visually or textually. Assuming a sum operator defined in the target language toolkit (see [15] for details), constraint *TotalBalanceIsPositive* (Fig. 7) is expressed in Z; its text is embedded in VCL CntD.

6 Discussion

VCL and our previous work. This paper presents part of our ongoing work on VCL, a visual language for abstract specification of software systems. VCL uses our previous result, ZOO [15,5], a semantic domain of object-orientation expressed in language Z, which is well studied; it has been applied to several case studies published in the literature. This enables us to focus on the visual aspects of VCL; a result of work presented here is that we can describe visually structures that previously could only be described textually in Z.

Use of Alloy. Metamodels of VCL notations presented here were formally defined in Alloy (see [14]), and refined into concrete syntax metamodels implemented in VCL's *visual contract builder* tool[4] (an Eclipse plug-in based on GMF framework[5]). Alloy was of great help in defining VCL's syntax: (a) it enabled

[4] http://vcl.gforge.uni.lu
[5] http://www.eclipse.org/modeling/gmf/

Table 2. Comparison of visual expressiveness in relation to generated Z between VCL model of case study's *Bank* package and UML-based model of [15,5]

	Total Lines of Z	**From visual**	**Percentage of visually**
With VCL	263	257	97.7%
With UML as in [15]	215	195	91%

precise expression of well-formedness constraints; (b) its model-finder and visualisation features helped in understanding VCL's syntax; (c) its model-checking feature verified satisfaction of certain desired properties; and (d) OO structure of its models meant a smooth transition from abstract to concrete syntax (VCL's tool uses OO metamodel-based technologies to construct graphical parsers).

Usability. This has been a concern guiding VCL's design:

- VCL's visual concepts are designed to be well-matched to meaning and give good sense of their mathematical underpinnings (*closeness of mapping* guideline of [17]). VCL's blob symbol, for instance, a circular contour denoting a set, is a well-known mathematical visual concept (as Venn or Euler circles).
- To enable users to infer meaning from patterns, VCL comprises a minimal set of primitives that have some core meaning, which varies slightly with the context (*consistency* guideline of [17]).

Expressiveness. VCL is designed to enable precise and rigorous abstract specification and to express visually constraints not visually expressible in UML. VCL's design is accompanied by an outline of a formal Z semantics, which has been illustrated here with examples; Z model representing semantics of case study's VCL model presented here is given in [16].

VCL was able to express visually 3, out of a total of 4, system invariants of case study; UML-based description of [15,5] describes none of them. Table 2 compares number of lines of generated Z for VCL specification presented here, and UML-based description of [15,5]. 97.7% of case study could be expressed visually using VCL; remaining 1.5% (constraint *TotalBalanceIsPositive*, above) must be expressed textually. This gives a 6.7% increase from [15,5].[6]

Modularity. VCL examples given in this paper highlight VCL's modularity and abstraction mechanisms. The constraint visual primitive abstracts away from the details of constraint definitions in CntDs. CntDs are modules that can be composed in various ways. For example, invariant *HasCurrentBefSavings* described in Fig. 4 is defined using two auxiliary CntDs, *CustsWithCurrentDef* and *CustsWithSavingsDef*, which are combined in CntD *HasCurrentBefSavings* through importing. The same auxiliary CntDs could be used to state other constraints which require the set of customers with a current account account, and the set of customers with a savings account. Invariant *CorporateHaveNoSavings*

[6] This increases to 54.4% with VCL's language of contracts; [15,5] describes very few behaviour visually.

of Fig. 5 illustrates how CntDs can be combined using logical operators, such as universal quantification and implication.

Two Modelling Styles. VCL enables two styles of constraint specification. Blob constructions enable specification based on sets and relations. Object-based constructions closely mimic predicate calculus formulas. Using either style is often a personal choice, but some solutions call for blobs, others for objects. In our experience, blob constraints tend to be more compact, but users familiar with predicate calculus find object expressions easier to follow. For instance, blob constraint of Fig. 3 (p. 267), is more compact than semantically equivalent object constraint of Fig. 5 (p. 268) because it uses fewer modelling elements. On the other hand, object constraint of Fig. 6 (p. 269) is more readable than that of Fig. 4 (p. 268) by those more familiar with predicate-calculus. Semantically, blob constraints tend to result in more concise and compact Z expressions.

Practical Value. We design a language for visual expression because, as argued in [1,2], there is value in them. VCL has been applied to a case study of a large-scale system [18]. We found that it was more productive to specify in VCL than in Z directly, and that VCL enhanced usability and readability of resulting specification.

7 Related Work

Evans et al [7] propose to define formally UML's semantics in Z; intent is to work on UML realm only. VCL's semantic approach generates a working Z model, which can be used for proof and animation by Z experts, and to support model analysis assisted by diagrams as proposed in [19,5]. As shown with case study, Z is also used to *augment* visual description and express what can not be expressed visually (constraint *TotalBalIsPositive*, Fig. 7, p.269), and VCL is able to express visually what was not visually expressible with UML.

Several approaches propose visual constraint notations to eliminate or minimise need for textual languages like OCL. These fall into two groups depending on supported style of constraint specification: *sets* and *predicate-calculus*. Constraint or spider diagrams [20,21], like VCL's blob constructions, are akin to Euler diagrams in that they express set-based constraints (inclusion, intersection, etc). Visual OCL [22] and Story Decision Patterns [23] have a semantics based on graph-transformation; they result in constraints akin to predicate calculus like VCL's object constraints. To our knowledge, VCL is the only visual language that integrates both styles of constraint specification. However, these languages are more mature than VCL, which still lacks a complete formal definition.

Another prominent feature of VCL is its support for modularity. Constraint diagrams [20,21] lack mechanisms to compose constraints modularly similarly to the VCL constraint composition mechanisms illustrated here. Visual OCL [24,22], like VCL, also provide logical operators and quantifiers, and a way of composing constraints, but does does not support set-based constraints; also

Visual OCL has more visual concepts than VCL, which does not favour usability. Story Decision Diagrams [23] notation has modular operators and quantifiers. In terms of expressability, [23] is close to our object language; when trying to express the constraints of [23] in VCL, structures used in both solutions were close to each other. However, VCL's syntax is closer to predicate calculus than [23] — VCL's logical operators are standard implication, conjunction, negation and disjunction (it isn't so in [23]) —, and so it benefits from engineers' familiarity with predicate calculus, and VCL enables specification of set-based constraints.

Our work is influenced by Harel's *Higraphs* [2], which are based on Euler diagrams and are basis of *statecharts*. From [2], we borrowed the *blob* and took inspiration for both language of VCL SDs and blob-based constructions of CntDs.

8 Conclusions

This paper presents some results regarding our ongoing work on VCL, a visual and formal language for abstract specification of software systems. It presents design of VCL's structural and constraint diagrams with a case study, and illustrates outline of VCL's formal Z semantics that accompanies VCL's design presented here by showing how VCL diagrams would be represented in Z.

VCL presented here is just a design of a language. This design includes an outline of the language's formal Z semantics. We intend to have a complete formal definition of VCL. Currently, we are working on defining formally the *semantic mapping*, from syntax to Z semantics illustrated here, which will be the basis of automatic generation of Z models in VCL's tool.

This paper demonstrates VCL's modularity at the level of constraints and its capability at expressing visually constraints not expressible visually in UML. It shows that VCL was able to express more visually than a UML-based approach for the case study used here. VCL is able to describe 3 out of 4 system invariants; UML describes none of them. The paper illustrates two styles of visual constraint specification: one is set-theoretic, the other is akin to predicate calculus.

We are working on a coarse-grained modularity mechanism of packages to enable separation of concerns at the requirements level [12]. We have successfully applied this mechanism to tackle complexity of a large-scale case study in [18].

There are several aspects in the work presented here that are, to our knowledge, novel. The modularity of VCL's approach to constraint specification is something not much explored in this area. Perhaps, the most relevant novelty is that VCL enables the specification of constraints visually in both set-theoretic and predicate-calculus styles. To our knowledge, no one integrated in a single constraint language Euler-like diagrams with object graphs used in graph transformation approaches.

References

1. Larkin, J.H., Simon, H.A.: Why a diagram is (sometimes) worth ten thousand words. Cognitive Science 11, 65–99 (1987)
2. Harel, D.: On visual formalisms. Commun. of the ACM 31(5), 514–530 (1988)

3. Sumner, M., Sitek, J.: Are structured methods for system analysis and design being used. J. of Systems Management 37(6), 18–23 (1986)
4. Anda, B., Hansen, K., Gullesen, I., Thorsen, H.K.: Experiences from introducing UML-based development in a large safety-critical project. Empirical Software Engineering 11(4), 555–581 (2006)
5. Amalio, N.: Generative frameworks for rigorous model-driven development. Ph.D. thesis, Dept. Computer Science, Univ. of York (2007)
6. Amálio, N., Polack, F., Stepney, S.: Frameworks based on templates for rigorous model-driven development. ENTCS 191, 3–23 (2007)
7. Evans, A., France, R.B., Lano, K., Rumpe, B.: The UML as a formal modelling notation. In: Bézivin, J., Muller, P.-A. (eds.) UML 1998. LNCS, vol. 1618, pp. 336–348. Springer, Heidelberg (1999)
8. Woodcock, J., Davies, J.: Using Z: Specification, Refinement, and Proof. PH (1996)
9. Jackson, D.: Software Abstractions: logic, language, and analysis. MIT Press, Cambridge (2006)
10. Cleland, G., MacKenzie, D.: Inhibiting factors, market structure and industrial uptake of formal methods. In: WIFT 1995, pp. 46–60. IEEE, Los Alamitos (1995)
11. Harel, D.: Biting the silver bullet: Toward a brighter future for system development. Computer 25(1), 8–20 (1992)
12. Amálio, N., Kelsen, P., Ma, Q.: The visual contract language: abstract modelling of software systems visually, formally and modularly. Tech. Report TR-LASSY-10-03, Univ. of Luxembourg (2010), http://bit.ly/9c5YwQ
13. Amálio, N., Kelsen, P.: VCL, a visual language for modelling software systems formally. In: Diagrams 2010. LNCS. Springer, Heidelberg (2010)
14. Amálio, N., Kelsen, P.: The abstract syntax of structural VCL. Tech. Report TR-LASSY-09-02, Univ. of Luxembourg (2009), http://bit.ly/4DHWky
15. Amálio, N., Polack, F., Stepney, S.: An object-oriented structuring for Z based on views. In: Treharne, H., King, S., C. Henson, M., Schneider, S. (eds.) ZB 2005. LNCS, vol. 3455, pp. 262–278. Springer, Heidelberg (2005)
16. Amalio, N.: ZOO specification of VCL model describing structural aspects of simple bank case study (2010), http://bit.ly/4yBrsW
17. Green, T.R.G., Petre, M.: Usability analysis of visual programming environments: a 'cognitive dimensions' framework. J. of Visual Languages and Computing 7(2), 131–174 (1996)
18. Amálio, N., Ma, Q., Glodt, C., Kelsen, P.V.: specification of the car-crash crisis management system. Tech. Report TR-LASSY-09-03, University of Luxembourg (2009), http://vcl.gforge.uni.lu/doc/vcl-cccms.pdf
19. Amálio, N., Stepney, S., Polack, F.: Formal proof from UML models. In: Davies, J., Schulte, W., Barnett, M. (eds.) ICFEM 2004. LNCS, vol. 3308, pp. 418–433. Springer, Heidelberg (2004)
20. Kent, S.: Constraint diagrams: Visualizing invariants in object-oriented methods. In: Proc. of OOPSLA 1997, pp. 327–341. ACM Press, New York (1997)
21. Fish, A., Flower, J., House, J.: The semantics of augmented constraint diagrams. J. of Visual Languages and Computing 16, 541–573 (2000)
22. Ehrig, K., Winkelmann, J.: Model transformation from visual OCL to OCL using graph transformation. ENTCS 152, 23–37 (2006)
23. Giese, H., Klein, F.: Beyond story patterns: Story decision diagrams. In: Proc. of Fujaba Days 2006, pp. 2–9 (2006)
24. Bottoni, P., Koch, M., Parisi-Presicce, F., Taentzer, G.: A visualisation of OCL using collaborations. In: Gogolla, M., Kobryn, C. (eds.) UML 2001. LNCS, vol. 2185, pp. 257–271. Springer, Heidelberg (2001)

Configuring the Variability of Business Process Models Using Non-Functional Requirements

Emanuel Santos[1,2], João Pimentel[1], Jaelson Castro[1], Juan Sánchez[2], and Oscar Pastor[2]

[1] Universidade Federal de Pernambuco, Centro de Informatica, Cidade Universitaria S/N, 50741-000 Recife, Brazil
{ebs,jhcp,jbc}@cin.ufpe.br

[2] Universidad Politecnica de Valencia, Camino de Vera, S/N, 46022. Valencia, Spain
{jsanchez,opastor}@dsic.upv.es

Abstract. The existence of variations in the organizational environment makes the configuration of business process models a complex activity, even for experienced business analysts. The increasing adoption of business processes models by software engineers as a input for requirements analysis strengthens the importance of adressing this issue. The challenge is to configure business processes to fit the organization better. We propose an approach that combines variability analysis and non-functional requirements to drive the configuration of a business process. Applying this approach we can analyze variability in the model in order to assess the impact of the choices on the process quality constraints - the non-functional requirements. Moreover, it provides a rationale for the selection of a specific configuration.

Keywords: Business Process Models, Business Process Configuration, Variability, Non-Functional Requirements.

1 Introduction

With the increasing interest of the software engineering community in using business process models as a source of requirements, raised the importance of representing variability on these models. Variability, on business process models, consists of defining alternative paths of execution in a workflow [1]. In this way, the process can be personalized for a specific context, e.g., for a foreign subsidiary of a corporation.

There are several approaches for representing variability in a business process model, like Schnieders and Puhlmann [2], Montero et al. [3] and La Rosa et al. [4]. However, the problem of choosing the most suitable alternative - the so-called process configuration - is not solved yet. In the industry, the configuration still is performed in an ad hoc basis, guided only by the analyst's experience. Some techniques have been proposed in academia, like the usage of questionnaires [4] and domain analysis [5], but these techniques are more concerned with the elicitation of variability than with the configuration itself.

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 274–286, 2010.

In this paper we propose an approach for performing business process configuration based on its non-functional requirements (NFR). These requirements, sometimes called quality requirements, define constraints that the process must comply to. We believe that non-functional requirements are a suitable criterion for guiding the process configuration, since they represent the high-level characteristics from which processes are usually evaluated - cost, performance, accuracy, and so on. Also, the solid foundations on which software systems NFR is built [6] provide plenty of techniques that can be borrowed and used in this new domain. Some recent works are already heading toward the integration of NFR and business process models [7] [8] [9].

We are going to present our approach using the Business Process Model and Notation - BPMN [10], since it is a well known and acknowledged notation in the software engineering community. However, this approach can be applied to any other process notation in which variability can be expressed.

In summary, the main contributions of this paper are twofold:

- The definition of an approach for business process configuration using non-functional requirements (NFR) as the selection criteria. We describe how to model variability in the business process (first phase), link the process variants to the NFR and use the linkages to select the best configuration for a selected NFR (second phase).
- The integration of current NFR techniques and algorithms, aiming to enable the automatic configuration of a business process. In this way, the configuration could be performed at design time, by a process analyst, or at run-time, by the system itself.

In Section 2 we are going to introduce the background of our research, namely BPMN and non-functional requirements. Following, we present the approach itself, in Section 3. The application of our approach is exemplified in Section 4. Related works are presented in Section 5. Finally, in Section 6, the conclusions are discussed.

2 Background

Business Process Model and Notation (BPMN) is a notation to model business processes in terms of their activities and supporting information [11]. The BPMN is based on the representation of activities flows and allow to represent different levels of details for different purposes. Figure 1 depicts the most commonly used BPMN elements and their graphical notation. In BPMN, the roles that participate in a process are represented by Pools and Lanes. Pools represent organizations and Lanes represent the participants or subdivisions of an organization. The process is composed by sub-processes and tasks, connected through flows of communication. There are elements that represent the events that start the process, that finish the process or that happen during the execution of process (i.e., intermediary events). With these elements the business analyst can represent, analyze and propose improvements to the business processes of an organization.

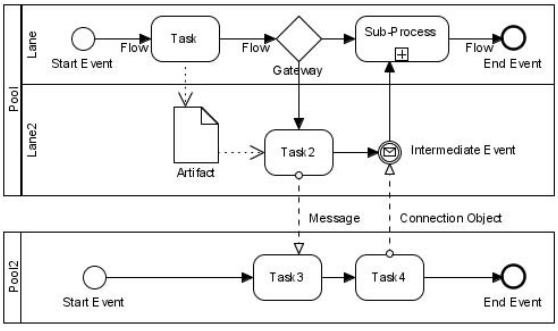


Fig. 1. BPMN elements

The non-functional requirements (NFR) are requirements that specify criteria to judge the operation of a software, rather than a specific behavior. It usually is represented in terms of qualities, or constraints, that the software should be concerned with. The NFR can be seen as properties observable at run-time - such as security or usability, or as properties embodied in the product - such as maintainability or scalability.

Chung et al. [6] describes a framework to model and analyze non-functional requirements. The NFR Framework is based on the concept of Softgoal that is a representation of non-functional requirements. The softgoals are characterized as the goals that the system should achieve but that have no clear achievement criteria. The NFR Framework uses the Softgoal Interdependency Graph (SIG) to refine and analyze the interation among softgoals using the decomposition and contribution analysis. The models generated using the NFR Framework can be grouped in reusable catalogues.

3 The Approach

Our approach aims to provide a way for configuring business process models, maintaining the rationale behind the selection of a specific process instance. It is divided in two phases, each one with two steps. The first phase consists of analyzing the business process model, which we are modeling with BPMN, in order to discover and represent possible variations of the model. In the second phase we analyze the non-functional requirements and perform the configuration itself.

We describe the variations in terms of variants and variation points [12]. The variation points are the subject of variation, in the case o BPMN could be tasks, events, artifacts, or pools. They are points where new elements can be added, replaced or removed to represent different behaviors of process. In its turn the variants describe the object of variation, for example, if a task could be performed in different ways each one could be represented as a variant.

3.1 Phase 1 - Elicitation and Representation of Variability in a BPMN Model

In this first phase we are going to elicit variability (Step 1.1) and represent it on the process model (Step 1.2), assuming that this have not been done yet. So, in this phase the aim is to identify and to organize the variations that can be found on a given business process.

Step 1.1 - Elicit variability. The elicitation of variability is the activity of identifying and discovering possible variations in a model. The goal is to identify different ways to carry on a process, what could result in the inclusion, changing or exclusion of elements on the model. To perform this elicitation it we use an information analysis framework [13] that explores different facets of the information and obtain new data about it. In the context of BPMN models we will use this framework to inquire the tasks, activities and sub-processes of model and identify new information about them. The use of this framework is as simple as making questions like Who? How? When?, which is very usual in requirements engineering.

The facets that can be identified, with the respective questions, are:

- Agentive (Who will perform the task?)
- Dative (Who will be affected by the task?)
- Objective (What are the objects consumed or produced by the task?)
- Extent (What are the degree of the task will be performed?)
- Process (How the action will be executed?)
- Conditional (In what conditions the task will be performed?)
- Locational (Where the task will be performed?)
- Temporal (When the task will be performed?)

Asking those questions to each element of the process model, we can identify a comprehensive set of possible variations on a business process. In the next step we are going to represent these variations in terms of the BPMN notation.

Step 1.2 - Describe variability. The elicitation results in a list of variations that need to be represented in order to reflect the nature of business process. So, in this step we put the variations in terms of the BPMN notation. In doing so we can apply these variations in the BPMN model while maintaining the consistence of the notation. As explained before, we represent the variations using the concepts of variation points and variants. Variation point is the place where the variation occurs, and each possible alternative for a variation point.

To describe the variants we are using an identifier, the point where should be inserted, the dependencies that can be present and a pattern of insertion. The patterns of insertion are already described in the literature [14]. They can be the insertion of sequences, parallelism, optional behavior, and so on. We identified that this set of patterns were too limited for this approach, so we complement them with patterns for deletion, insertion of lanes and substitution. The deletion pattern covers the case of a negative dependency that happens when a variant

excludes another variant. In this pattern all elements in the variation point are deleted and the beginning and the end of the variation point are linked directly. The insertion of a lane is a pattern specific for BPMN and is related to the inclusion of new roles in the model.

Variation points are described with an identifier (name), a type (task, link, sequence), a point of reference (begin and end), and a list of the variants that can be placed in it. Throughout the example of Section 4 we present some examples of variation points and variants.

3.2 Phase 2 - Analysis and Configuration

The variation points and the variants are essential inputs for performing the process configuration. However, this information by itself is not sufficient to identify which subset of variants results on the best process. So, in this phase we are going to link the variants to non-functional requirements and analyze which process configuration maximize a selected criterion.

Step 2.1 - Link variants to NFR. In this step the non-functional requirements (NFR) will be linked to the business process variants identified earlier. To do so, we first need to identify which NFR will be taken into consideration. This can be done interviewing people involved in the business process [9], using requirements catalogs [7] or with a mix of elicitation techniques.

Once the NFR are identified, we will perform the linking between the process variants and the requirements. These links will be represented using matrices, which is a usual and scalable solution for representing this kind of information. Moreover, matrices allow the building of views containing only a partial representation of the variants and the requirements, simplifying its analysis.

In Table 4 (Section 4) we provide a example of the variants to NFR matrix. The lines of the matrix are the process variants, grouped by its variation points, and the columns are the non-functional requirements. For each variant, we are going to define the impact of that variant on each NFR, in a scale of -1 to 1 (inclusive). A negative value in this scale means a negative impact, as well as a positive value means a positive impact. Zero is the neutral value, meaning that variant does not impact the NFR at all.

In order to make it more user friendly, this scale can be replaced by any other scale, provided that the required transformation is performed. For instance, let us consider the qualitative scale of the NFR Framework [6]. In the NFR Framework, the most positive impact on a non-functional requirement is *Make*, a partial positive impact is *Help*, a partial negative impact is *Hurt* and the most negative impact is *Break*. These values can be mapped, respectively, to 1, 0.5, -0.5 and -1, in our scale.

Now that we have the linkages between the process variants and the NFR, we can perform the configuration itself, in the next step.

Step 2.2 Perform the configuration. At this point we know the variation points and the variants of the business process, and how they impact the

non-functional requirements. Now we will use this data to support the configuration itself.

There are two possible ways for analyzing the impact of each configuration on the NFR: top-down analysis and bottom-up analysis. In the top-down analysis we select which non-functional requirement has the maximum priority, and then derive a process configuration that maximizes the selected NFR. Alternatively, in the bottom-up analysis we define a process configuration, by selecting a subset of variants, and then observe how this configuration affects the non-functional requirements.

These analysis can be performed semi-automatically. The algorithms to perform the evaluation of alternatives using non-functional requirements are already available in the literature [6] [15]. The choice of matrices as data structure allows the usage of even more sophisticated algorithms, in order to resolve dependencies and conflicts that may arise. However, it is up to the analyst to select the NFR used as criteria - in the top-down analysis - or the configuration that will be evaluated - in the bottom-up analysis.

Following, we present the analysis themselves:

Top-Down Analysis. The top-down analysis consists of obtaining an instance of the model based on the selection of a non-functional requirement. So, the analyst will define which NFR will be prioritized. Each variation point is evaluated to identify the variant that better fits the selected non-functional requirement. I.e., the variant which has the biggest positive impact on that NFR. This evaluation can be performed automatically. However, dependencies between variants have to be taken into consideration as well. If a variant X require the variant Y, the calculation will be performed considering X and Y altogether.

Bottom-Up Analysis. The bottom-up analysis consists of selecting a subset of variants and using the linkage matrix to calculate the impact of that configuration on the non-functional requirements. This way an analyst could, for instance, evaluate if the current configuration is satisfactory.

A good way of performing the configuration is to perform a top-down analysis and then evaluate subtle changes of the configuration, using bottom-up analysis. This way the analyst will have a starting point for the configuration and will be able to understand how his changes affect the non-functional requirements. At the end, the analyst will have not only the process instance, but also the rationale for choosing that instance. E.g., "this configuration maximizes the accuracy, while maintaining a low cost".

4 Running Example

In order to demonstrate the application of our approach, we will introduce an example of Conference Management. During the organization of a conference several activities are realized: the call for papers, the revision of papers, the organization of proceedings, and so on. The diagram on Figure 2 presents an excerpt of a process of revision and notification of acceptance in a small conference.

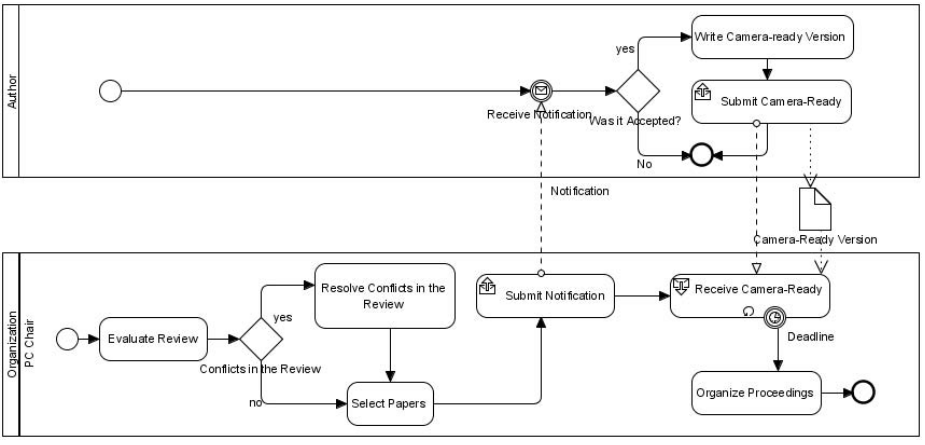


Fig. 2. CMS example

In this process, an Author has previously submitted a paper for a conference and is waiting for the results of the evaluation of his paper. In the conference committee, the PC Chair, which organizes the conference program, is responsible for evaluating the papers reviews and deciding if the paper will be accepted or rejected. Due to space problems we describe only the variants of the task Submit Notification. Due to space problems we describe only the variants of the task *Submit Notification*.

We start with the Step 1.1, performing an analysis of information facets [13] in the task Submit Notification. The task Submit Notification can be performed by the PC Chair or, automatically, by a Conference Management System (CMS). The reception of the notification can include all the authors or be directed just to the first Author. The notification can be sent by email or using a CMS. Finally, the notification submission can be done when a deadline arrives or when all reviews are collected. The identified variations are shown in Table 1.

Analyzing the variations, we will identify what parts of the process will need to be modified to implement each variation - the variation points. We represent the

Table 1. Variants identified for *Submit Notification*

Task	Facet	Variants
Submit Notification	Agentive	PC Chair
		CMS System
	Dative	First Author
		All Authors
	Process	By E-mail
		By publishing in CMS
	Conditional	When the deadline finish
		When all revisions are available

Table 2. Variation Points

Variation Points				
ID	Type	Begin	End	Variants
1	Task	Submit Notification	Submit Notification	1,2
2	Link	Select Papers	Submit Notification	3,4
3	Lane	Author	Author	5,6
4	Lane	PC Chair	PC Chair	5,8

variation points as in Table 2. For each variation point we need to define where they begin and end. These are the points where the variations can be placed. The variation points listed in the Table 2 are the task *Submit Notification* itself, the lanes of *Author* and *PC Chair*, and the link between *Select Papers* and *Submit Notification*.

The variants represented in Table 3 are the same from Table 1, but now with the specification of the type of variants, the dependencies, and the patterns of insertion. In this example, the variant 2 requires the selection of the CMS System as the agent of the task *Submit Notification*, i.e., the variant 2 has a dependency to the variant 5. In the variants 1, 2 and 7, there is a need to change the name of a task, or to replace a task with another one, so their pattern is substitution. The task *Submit Notification* can be substituted by *Submit Notification by E-mail* or *Submit Notification by posting in CMS*.

Now that we know the possible variations in the business process, we are going to define the linking among variants and non-functional requirements. For the sake of space, we are going to consider just two non-functional requirements: Cost and Availability. The aim is to minimize the cost of applying a solution and maximize the availability - i.e., the capacity of readily provide information to the participants of the process. The values of the contribution links varies in a scale from -1 to 1, that means from a negative contribution (increase the cost or damage the availability) to a positive one (minimize cost or maximize

Table 3. Representation of variants

Variants					
ID	Name	Type	Pattern	Dependencies	Variation Point
1	Submit Notify by e-mail	Task	Substitution		1
2	Submit Notify by posting in CMS	Task	Substitution	5	1
3	Deadline	Time Event	Insertion		2
4	All revisions are available	Time Event	Insertion		2
5	CMS System	Lane	Insertion		4
6	Collaborators	Lane	Insertion		3
7	First Author	Lane	Substitution	6	3
8	PC Chair	Lane	Maintain		4

Table 4. Variants and the relationships with Non-functional requirements

Variation Point (ID)	Variants	NFR	
		Cost	Availability
4	PC Chair	0	0
	CMS System	-1	1
3	First Author	0.5	-1
	All Authors	0	0.5
1	By E-mail	0	0
	By publishing in CMS	-1	1
2	When the deadline finish	0	0
	When all revisions are available	0	0

availability). Table 4 presents the result of the linkage between the variants and the non-functional requirements. The values assigned for each variant reveals the impact on Cost and on Availability of the process. For instance, the selection of a CMS as the agent that will perform the notification submission requires the development of a system, which increases the cost of this process. On the other hand, it presents benefits on the process availability, by provinding an accessible environment to share information.

We used the top-down analysis to obtain the instances of the Conference Management process presented in the figures 3 and 4. Figure 3 shows a process configuration that prioritizes Cost, over Availability. It presents the *PC Chair* as responsible for executing the notification submission, and the selected means for doing so is by e-mail. Other variants such as the *Deadline* were included even being neutral if compared with the other variants of the same variation point. The configuration prioritizing Availability (Figure 4) uses some variants that could not be included in the configuration that prioritizes Cost.

4.1 Discussion

The usage of information facets allows a quick elicitation of variability, in contrast to approaches like questionnaires and domain analysis. This is due the pre-defined, objective and limited set of questions that need to be answered during the analysis of information facets.

Our approach is part of an ongoing work. In this way, it may present some limitations. The application of our work may show to be too time consuming, since for every element in the business process we may identify several variations. This effort is multiplied by the number of non-functional requirements being considered. However, this seems to be an inherent problem of any approach that deals with variability, since the amount of variations that may arise in real situations is potentially large. Moreover, we believe that improvements on our approach, for instance, the automation of some of its steps - can minimize this problem. Another limitation is that our approach requires the analyst to have a high expertise on the domain of the process being modeled and to be familiar with the BPMN notation.

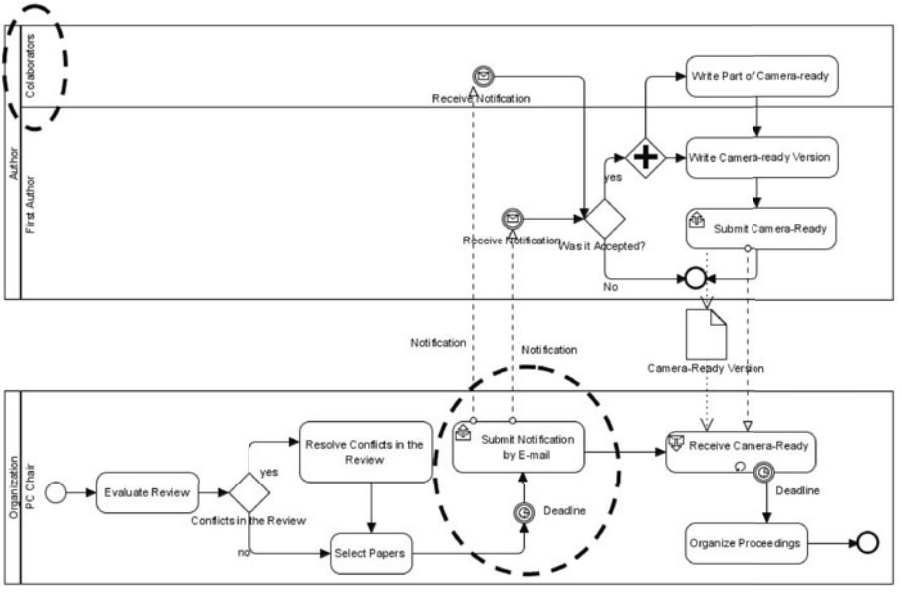


Fig. 3. CMS example with configuration prioritizing Cost

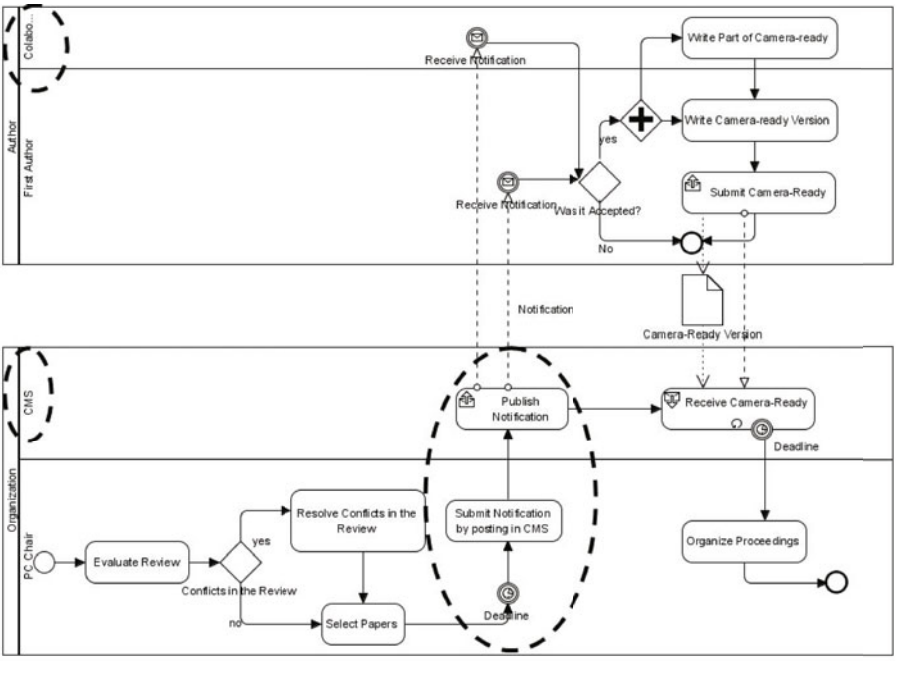


Fig. 4. CMS example with configuration prioritizing Availability

5 Related Work

Schnieders and Puhlmann [2] present a mechanism to represent high variability business process models using BPMN. In this approach they present mechanisms to represent variability in flow-based languages. They rely on extension, inclusion, parameterization and design patterns. These mechanisms enrich the BPMN model and allow the representation of variability with a specific representation for each type of variability. They propose using feature models to obtain the variability but do not explain how to do it. Moreover, their approach is focused on the process itself, without consider the requirements phase. Our approach is not concerned with the representation of variability in the BPMN model itself as Schnieders and Puhlmann do [2]. We believe that the variability represented in an independent model helps the readibility of model.

In Lapouchinian et al. [5] there is an approach that represents business process in terms of its goals. Variability rich business processes are modeled using goal graphs. As the goal graphs are not expressive enough to represent flow and sequence, they apply annotation in the model in order to cover this gap. The aim of this approach is obtain configuration mechanisms that reflect the business process. The result is a configuration mechanism that abstracts the complexity of configuring software from the end-users. Their approach can generate business process (described in BPEL) based transformation of the goal model. We intent to use the non-functional requirements to drive the configuration of models such as Lapouchinian et al. [5]. However by using a generic structure to represent the variability (i.e., matrices) we avoid the work to deal with two types of models (goal and business process models).

Montero et al. [3] describe a methodology to obtain and represent variability in business process models, represented by BPMN language. They are concerned with the derivation of requirements for software related the business process. To represent the business process they adopt feature models and use cases model to describe requirements. The selection mechanism is the selection of features, then if a feature needs to be present in the solution it is selected and the model is restructured to support the changes. As formalism to do it they adopt finite state machines. They select the elements that will be part of the instance by selection of features, we proposed a similar strategy (using bottom-up configuration) but we also allow the configuration using the top-down strategy. Moreover, Montero et al. do not explain why an instance of the business process was selected as we do.

6 Conclusion and Future Work

In this paper we have presented an approach to guide the configuration of business process models using non-functional requirements. This approach spans from the elicitation of variability to the configuration itself, in which instances of the original model are produced. Besides guiding the configuration with clear criteria, this approach also provides the rationale for the selected configuration. In the running example we derived two instances of a conference management

process, each one prioritizing a different NFR: the first one was the instance that resulted in the lower cost, and the second one provided the higher availability.

We consider that the hardest part of this approach is defining the degree of impact of each variant on the NFR. This could be softened through the creation of a catalog that suggests, for each kind of activity in a business process, the impact that activity has on a list of non-functional requirements. This approach can be considered an early activity of the requirements engineering phase, since the resultant process configuration can be used to identify the requirements of an information system to support that process [16]. Since the top down analysis of the configuration can be performed automatically, the system itself could perform a reconfiguration considering context changes that arise during its execution. This behavior is classified as the second level of requirements engineering in dynamic adaptive systems [17].

As future work, we expect to implement supporting tools for this approach. We also intend to improve the priorization used in the top-down analysis, allowing more than one non-functional requirement to be used as criteria, with different weights. Lastly, we are planning to validate this approach performing experimentation with more complex processes. The related works presented in section 5 have parts that are similar or equivalent to parts of our approach. Even if we can not compare the whole approach, we still could compare the similar parts.

Acknowledgments

This work has been partially funded by CNPQ Procs. 143185/2008-0, 565349/2008-2, 308587/2007-3, FACEPE (Grant IBPG-0173-1.03/08), and Project CAPES/DGU Proc. 167/08.

References

1. Halstead, M.H.: Elements of software science. Operating, and Programming Systems Series 7 (1977)
2. Schnieders, A., Puhlmann, F.: Variability Mechanisms in E-Business Process Families. In: Proceedings of 9th International Conference on Business Information Systems, BIS (2006)
3. Montero, I., Peña, J., Ruiz-Cortés, A.: Representing Runtime Variability in Business-Driven Development Systems. In: Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008), pp. 605–608. IEEE Computer Society Press, Los Alamitos (2008)
4. La Rosa, M., van Der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: Questionnaire-based variability modeling for system configuration. Software and Systems Modeling 8(2), 251–274 (2009)
5. Lapouchnian, A., Yu, Y., Mylopoulos, J.: Requirements-Driven Design and Configuration Management of Business Processes. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 246–261. Springer, Heidelberg (2007)

6. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering, vol. 5. Kluwer Academic Publishers, Dordrecht (2000)
7. Cardoso, E.C.S., Almeida, J.a.P., Guizzardi, G., Guizzardi, R.S.S.: Eliciting Goals for Business Process Models with Non-Functional Requirements Catalogues. In: Enterprise, Business-Process and Information Systems Modeling, pp. 33–45. Springer, Heidelberg (2009)
8. Pavlovski, C.J., Zou, J.: Non-functional requirements in business process modeling. In: APCCM 2008: Proceedings of the fifth on Asia-Pacific conference on conceptual modelling, pp. 103–112. Australian Computer Society, Inc. (2008)
9. Aburub, F., Odeh, M., Beeson, I.: Modelling non-functional requirements of business processes. Information and Software Technology 49(11-12), 1162–1171 (2007)
10. OMG, C.O.: Business Process Model and Notation (BPMN) - Specification v1.2 (2009)
11. White, S.A., Miers, D.: BPMN Modeling and Reference Guide: Understanding and Using BPMN. Future Strategies Book Division (2008)
12. Pohl, K., Bockle, G., Linden, F.V.D.: Software product line engineering, vol. 49. Springer, Heidelberg (2005)
13. Liaskos, S., Lapouchnian, A., Yu, Y., Yu, E., Mylopoulos, J.: On Goal-based Variability Acquisition and Analysis. In: Proceedings of 14th IEEE International Conference Requirements Engineering, RE 2006, pp. 92–96 (2006)
14. Wohed, P., van Der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M., Russell, N.: Pattern-based Analysis of BPMN (2005)
15. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Reasoning with goal models. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, pp. 167–181. Springer, Heidelberg (2002)
16. De La Vara, J.L., Sanchez, J., Pastor, O.: Business Process Modelling and Purpose Analysis for Requirements Analysis of Information Systems. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 213–227. Springer, Heidelberg (2008)
17. Berry, D.M., Cheng, B.H.C., Zhang, J.: The four levels of requirements engineering for and in dynamic adaptive systems. In: 11th International Workshop on Requirements Engineering Foundation for Software Quality (REFSQ), p. 5 (2005)

A Business Process Metadata Model for a Process Model Repository

Mturi Elias, Khurram Shahzad, and Paul Johannesson

Department of Computer and Systems Science (DSV),
Stockholm University (SU) / Royal Institute of Technology (KTH), Stockholm, Sweden
{mturi,mks,pajo}@dsv.su.se

Abstract. Today reuse of business process models is becoming increasingly important. One of the proven solutions for reusing business process models is the use of repositories. Repositories should have process models and process metadata that can help users in searching, understanding, and interpreting process models. The purpose of this paper is to propose a Business Process Metadata Model (BPMM) that would facilitate a) locating process models, b) understanding and/or interpreting process models, and c) navigating a process model repository. In order to evaluate the BPMM, an empirical study is conducted to measure consistency and correctness of annotating business processes by using BPMM.

1 Introduction

Business Process Management (BPM) has become one of the most important instruments that help modern organizations meet their business goals and achieve competitive advantage. Business process modeling plays a vital role in BPM. Motivations for modeling business processes include documenting current business processes, redesigning and improving processes, aligning business and IT, etc. While modeling of business processes remains a complex, costly and time consuming task [1, 2, 3], the efforts made to model business processes are seldom reused beyond their original purpose. Reuse of process models can reduce the cost and complexity of modeling business processes from scratch [3, 4].

A business process model repository is one of the proven approaches for supporting process models reuse [5, 7]. The repository provides a central location for storing, managing and changing process knowledge (business rules, relationships, process elements, etc.) [5, 6]. In addition, a repository enables stakeholders to retrieve process models for various purposes like understanding, updating, simulating and analyzing process models.

Reuse of process models cannot be done literally because it involves searching the process repository to find suitable models that can be the base for a new design. Therefore, the stored process models must be well described and classified to facilitate searching and interpretation. It has been argued that the use of process metadata and/or business context can meet these requirements [7, 8, 10, 11, 27]. Also, recent studies [9, 10] affirm that characterizing business processes facilitates understanding and navigation. However, only well structured metadata can increase the likelihood to

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 287–300, 2010.

properly understand and reuse business processes. Therefore, the aim of this study is to propose a business process metadata model (BPMM) for annotating business processes to facilitate locating, interpreting process models and navigating the repository.

The remainder of the paper is organized as follows. In section 2, we present the method used to develop BPMM, followed by a detailed description of the model. In section 3, we introduce an empirical study to evaluate BPMM by measuring annotation consistency and correctness. Finally, in section 4 the lessons learned and limitations of the study are presented.

2 The Business Process Metadata Model (BPMM)

In this section, we present the Business Process Metadata Model (BPMM) that can be used for annotating process models in the process model repository. The BPMM intends to facilitate: a) locating process models, b) understanding and/or interpreting process models, and c) navigating a process model repository.

The BPMM has been developed based on a systematic approach, which consists of three phases, identification of process related concepts, validation of the concepts and the model construction. Due to space limitations, we briefly describe the phases and the results.

Identification of Process related Concepts. In this phase a set of process related concepts (in this paper they are also referred as concepts) were collected by considering established business frameworks, process classification schemes and business process perspectives as inputs to an analysis and subsequent synthesis. The identified concepts include, a) *process description* b) *business context* as defined by [12], c) *business goal* from business process perspectives [13, 14], d) *domain specific classification scheme* based on [10, 16], e) *generic classification scheme* based on [10, 16], f) *process property,* g) *resource* and h) *actor* from REA [17, 18] and the process design framework [19], and i) *process relationship* [16].

Validation of Concepts. In this phase the identified concepts were validated through an empirical study which involved 25 volunteer participants. The participants included researchers and practitioners who participated in the *2nd Working Conference on the Practice of Enterprise Modeling* (PoEM'09) [20]. The participants were asked to assess whether annotating business processes with these concepts would facilitate searching, navigating and interpreting process models in a repository. A scale of 1 to 5 (from strong disagree to strongly agree) was used to validate the concepts. The results of the study are shown in fig. 1. The figure shows that a large number of participants agree (either agree or strongly agree) with most of the concepts.

The Model Construction. In this phase corresponding metadata elements of the validated concepts were defined. This was followed by defining relationships between elements and a business process.

Business context, *goal*, *resources*, *actors* and *process relationship* from the validated concepts were directly included as elements in BPMM. From the generic classification scheme (a concept) the following elements are defined: *process area* (based on Porter value chain [21]) and *process phase* (based on Open-EDI [22]). In addition

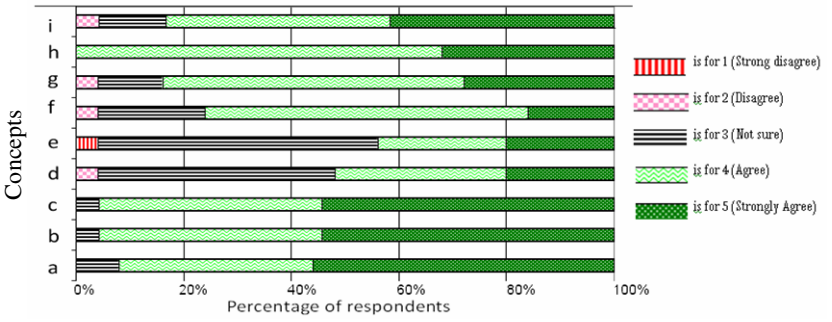


Fig. 1. Validation of Concepts

to that, we have introduced two more elements that do not have direct relationships with any validated concepts. However, these elements were derived from the participants' feedback and their references to different studies [17, 18, 23]. These elements *are process type* (based on REA framework [17, 18]) and *process level* (based on Organizational Theory [23, 24]).

Domain specific classification schemes were not included because the repository is intended to include processes which are not limited to a specific domain. Furthermore, widely accepted classifications for several domains are not available, e.g. a classification scheme for supply chain (SCOR [25]) is available, whereas an equally accepted classification for healthcare does not exist. Furthermore, properties of processes were not included in the model, because they vary between stakeholders.

The BPMM, therefore, includes *process type*, *resource*, *actor*, *process area*, *process phase*, *process level*, *business context*, *process relationship* and *goal* as shown in fig. 2. In the following two subsections we describe the elements and their relationships.

2.1 BPMM Elements

In this section, we describe each element of BPMM and discuss its purpose. The central component of BPMM is a business process which is to be annotated by the elements.

Process Area. The *process area* is based on the Porter Value Chain [21]. In order to better understand the activities through which an organization creates value, business processes are separated into areas. The process area element classifies business processes by their function or core competence. Therefore, annotating processes with process area enables users to identify business processes based on functional area. The process area can either be primary or supporting.

The primary process areas include:

- *Inbound logistics*: A process that includes activities needed for receiving, storing, inventory control, or transportation scheduling.
- *Operations*: A process that includes activities needed for value creation that transforms inputs into outputs. These include machining, packaging, assembly, equipment maintenance, testing and all other value-creating activities that transform the inputs into the final product.

- *Outbound logistics*: A process that includes activities required getting the finished product to the customers: warehousing, order fulfillment, transportation, and distribution management.
- *Marketing & Sales*: A process that includes activities associated with getting buyers to purchase the product including channel selection, advertising, promotion, selling, pricing, retail management, etc.
- *Service and Maintenance*: A process that includes activities that maintain and enhance the products value, including customer support, repair services, installation, training, spare parts management, upgrading, etc.

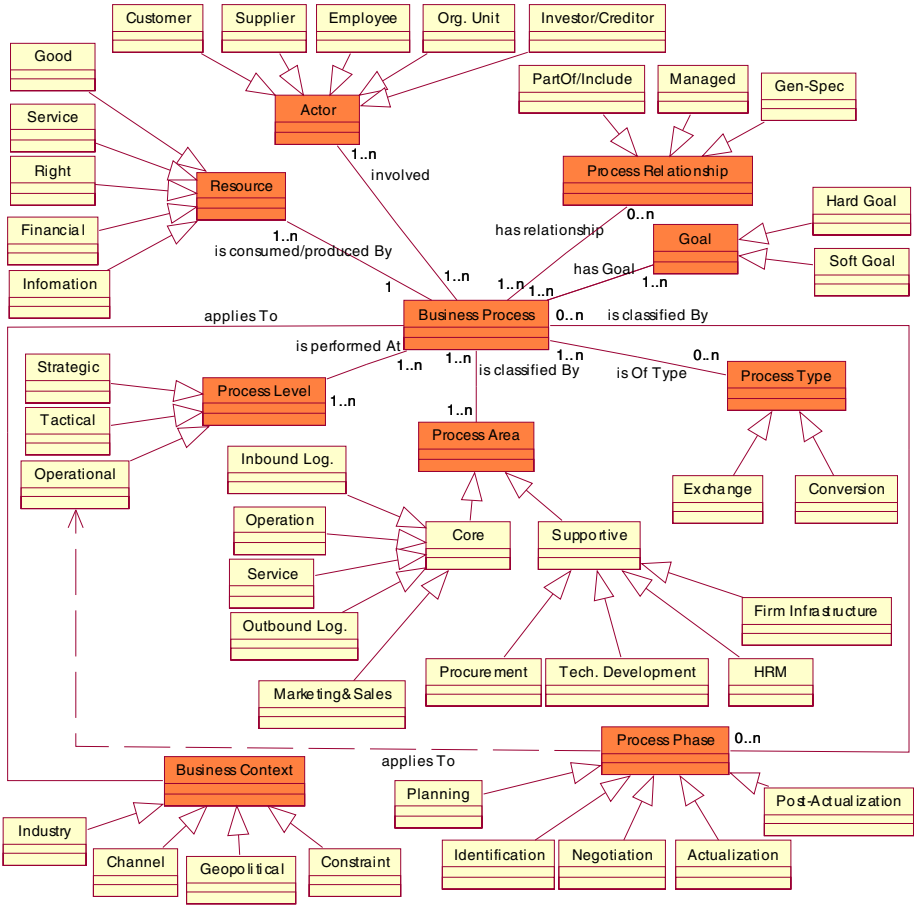


Fig. 2. The Business Process Metadata Model (BPMM)

The supporting process areas include:

- *Procurement*: A process that includes activities needed for acquiring raw materials, services, spare parts, buildings, machines, etc. These include information gathering on resource needs and supplier offerings, supplier contacts, background reviews on the quality of supplier offerings, negotiation, fulfillment, and supplier performance evaluation.
- *Technology Development*: A process that includes activities that support the value chain activities by developing new technology and procedures, such as Research and Development, Process automation, design, and redesign.
- *Human Resource Management*: A process that includes activities associated with recruiting, development (education), retention and compensation of employees and managers.
- *Firm Infrastructure*: A process that includes activities related to general management, planning management, legal, finance, accounting, public affairs, quality management, etc.

Process Phase. A business process can be constructed from a set of operational activities. In order to serve as a basis for co-ordination of work between different partners (in business collaboration) open-EDI [22] classifies these activities into five phases. The *process phase* element is based on the open-EDI. It defines the phase(s) to which a business process belongs. Therefore, annotating business processes with this element enables users to identify business processes based on the phases. The process phases are:

- *Planning*: The planning phase includes all activities needed to decide what actions to take for acquiring or selling goods and services. Here actors are concerned with the question of what goods or services to acquire or sell.
- *Identification*: The identification phase includes all activities needed to identify, select, and establish linkages with partners that are involved in the business collaboration. The question is with whom to do business.
- *Negotiation*: The negotiation phase includes all activities needed to establish a contract (agreement) and related commitments for the exchange of goods and services.
- *Actualization*: The actualization phase includes all activities needed to prepare and perform the resource exchanges stipulated in the contract established in the negotiation phase.
- *Post-Actualization*: The post-actualization phase includes the follow-up activities of resource exchanges performed in the actualization phase, e.g. warranty coverage, complaint handling, and after-sales service.

Process Type. Resources are produced and/or consumed through a series of business processes. The *process type* element classifies business processes around the resource life cycle, i.e. from acquisition, conversion to delivery of goods and services. The process type is based on the REA framework [17, 18]. Therefore, annotating processes with process type enables users to identify business processes based on the operations performed on a resource. The process types are:

- *Exchange*: a process in which an enterprise receives resources from another actor, and it gives resources to the other actor in return. The aim of such processes is to acquire, maintain, and pay for the resources needed by the organization as well as to sell and deliver goods and services to customers and collect payment. An example is the "Sales Process".
- *Conversion*: a process in which an enterprise uses or consumes resources in order to produce new or modify existing resources. The aim of such processes is to convert the acquired resources into goods and services for customers. The raw inputs are transformed into finished goods and services by this process. An example is the "Manufacturing Process".

Process Relationship. In order to achieve business goals, organizations perform several business processes that are related with each other. The *process relationship* element describes how business processes are related. Annotating business processes with relationship information helps users to identify related processes. Furthermore, relationship information supports traceability and process change management. The process relationship may exist in the following forms:

- A *generalization-specialization* relationship exists if one process (called a specialization) is a kind-of (or *is-a*) another process (called a generalization). Example "*Manage returns" is a generalization of "Manage returns with prior approval" and "Manage returns without prior approval",* while the latter two are specializations of the former.
- A *partof-includes* relationship exists if one process is composed of one or more processes (called sub processes). The sub process has the *partof* role and the parent process has the *includes* role. Example "Manage order approval" *includes* "Handle rejected order" and the later is a *partof* the former.
- A *manage/managed* relationship exists if one process plans, controls, monitors, evaluates, and/or designs another process.

Process Level. Levels are introduced in organizations in order to allow efficient management and coordination. Most organizations operate at three levels: strategic, tactical and operational [23, 24]. A *process level* element describes the level in the organization at which a business process is performed. Therefore, annotating business processes with process levels enables users to identify business processes based on organizational levels. The process levels are:

- *Operational*: A business process is said to be at the operational level if it includes activities that are performed on a day-to-day basis. The aim of such a process is to modify and exchange economic resources.
- *Tactical*: A business process is said to be at the tactical level if it includes activities that are performed on a short term plan. The aim of such a process is to manage operational level processes.
- *Strategic*: A business process is said to be at the strategic level if it includes activities that are performed on a long term plan. The aim of such a process is to define process types at the operational and tactical levels as well as the resource types to be used and produced.

Resource. A business process consumes, produces or transfers a resource between actors [17]. A r*esource* is anything (with or without physical substance) that is regarded as valuable by some *actors* [18]. This element is inspired by REA [17, 18] and the process design framework [19]. Therefore, annotating business processes with resources will enable users to identify processes based on the type of the resource consumed, produced or transferred by a business process. In most cases, a resource fits into one of the following categories:

- *Goods*: These are the physical or tangible objects (like cars, refrigerators, and cell phones) that are of value for an enterprise.
- *Services*: These are the non-tangible resources offered by actors to increase the value of some other resources. Example haircuts, eye treatments.
- *Rights*: These are eentitlements or permissions to an actor, usually of a legal or moral nature, i.e. ownership rights, usage rights, copyrights.
- *Financial*: These are funds or money (in the form of cash, cheque, voucher, credit card, etc) paid or received by an actor for goods or services being exchanged.
- *Information*: These are data in a certain context, like blueprints, referrals, and customer databases.

Actor. In the execution of a business process one or more actors may be involved [17]. An *actor* is an entity such as a person or an organizational unit involved in the realization of a business process. This element is inspired by REA [19, 18] and the process design framework [19]. Therefore, annotating business processes with actors will enable users to identify business processes based on the type of actor. In most cases an actor can fit into one of the following categories:

- *Customer*: An individual, company or organisation that buys goods or services.
- *Supplier*: An individual, company or organisation that provides goods or services to a recognisable customer or consumer.
- *Employee*: An individual who provides labour to an organization or another person.
- *Investor or Creditor*: A person, company, or entity that puts money or assets into an investment to yield returns.
- *Organization unit*: A subdivision or department in an organization/enterprise that is involved in a business process.

Business Context. 'In practice, one and the same business process varies a little bit with respect to the business environment' [26]. In order to reuse a process model, users may need to understand the business environment in which it is aimed to work. According to [26], a business environment can best be described by the concept of business context. A business context defines the circumstances in which a business process may be used [12]. This element enables users to identify business processes which may only apply to a specific business environment. The context in which a business process takes place can be specified by a set of categories and their associated values [12]. In BPMM we define the following contextual categories:

- *Industry*: provides the description of the industry in which the business process takes place.
- *Communication channel*: provides a description of the channel through which involved actors communicate.
- *Geopolitical*: provides a description of aspects related to region, nationality, or geographically based cultural factors.
- *Official Constraints*: describes those aspects of the business situation that result from legal or regulatory requirements.ï

Goal. The purpose of a business process is the achievement of one or more goals. A goal is a condition or state of affairs in the world that the actor would like to achieve [13]. In order to reuse a process model, users may need to understand whether a process model achieves their business goals. The *goal* element describes the business goals which a process model is aiming to achieve. Therefore annotating business processes with goals will enable users to identify a process model based on business goals. According to [14, 15] there are two types of goals associated with a business process:

- *Soft-goals*: these are strategic goals which are more abstract objectives that an organization is striving to achieve. For example the soft goal for the "procurement process" could be "minimize procurement costs".
- *Hard goals*: these are operational goals which define the state to be reached by a process (e.g. "complete an order").

3 Empirical Evaluation of BPMM

In this section we describe an experiment we have carried out for empirically evaluating the BPMM. Specifically, the purpose of the experiment is to evaluate consistency and correctness of annotating business processes using BPMM. Furthermore, the user perception of the model is tested.

3.1 Selecting Participants

The participants involved in the experiment were a mix of masters students in Engineering and Management of Information Systems (EMIS) and PhD students in Information Systems at KTH. By the time the experiment was done, all students had completed a course on Enterprise Systems and Modeling, in which they learnt basic concepts about business process modeling. The benefit of using student participants is that they form a homogeneous group with respect to their academic background and industrial experience. Furthermore, the experimental tasks did not require high level of industrial experience which justifies our selection of the participants.

3.2 Preparing the Experiment

For the experiment, the following materials were prepared:

- A document defining the BPMM model and the description of each element (as presented in section 3),

- A document describing (five) business processes. In order to increase the understanding, processes were presented in both textual and graphical form. Annotating business process is a time consuming task, therefore to keep the participants positive to the experiment, we had to limit the number of processes to five. The decision of limiting the business processes was also based on our experience from the pilot study (described below).
- A template for annotating business processes. It is a two dimensional table in which rows represent elements of the BPMM, and columns represent the processes to be annotated.
- A post task survey questionnaire to measure user perception of the model on a scale of 1 - 5 (Strongly Disagree, Disagree, Not Sure, Agree and Strongly Agree).

As part of the preparation of the experiment, a pilot study was conducted with three participants (PhD students). The purpose of the pilot study was to evaluate how well the participants were able to perform the experiment. The results and comments from this study were used to improve the BPMM elements definitions, business process descriptions and the template for the experiment.

3.3 Conducting the Experiment

For the experiment, 30 participants were given the materials (as described in section 3.2). This was followed by an explanation of the model, how to use the model and the template to annotate a process. Participants were then asked to annotate the business processes without any time constraint. After annotating the business processes, participants completed the post task survey. The response from 20 participants was received making the response rate 66.7%.

3.4 The Studied Variables

In order to evaluate the consistency and correctness of annotating business processes using BPMM, and the user perception of the model, the following three variables were defined:

Variable 1. *Annotation Consistency (AC)*: It is the degree to which process annotation (using BPMM) by different people is identical. AC is measured by the number (in percentage) of participants with identical process annotation on individual elements of BPMM.

The steps taken for measuring AC are to let different participants annotate a set of business processes and then we compute AC as follows:

1. Let $Max_{e,p}$ be the maximum number of participants with identical annotations on element *e* for process *p*. *e* is an element of {*Resource*, *Actor*, *Process Level*, *Process Relationship*, *Process Area*, *Process Phase*, *Process Type*}.

   For example, suppose a process (*p=1*) is annotated by 20 participants and for an element (*Process Level*), out of the 20 participants 12 annotate it as 'operational', 5 as 'tactical', and 3 as 'strategic'. Therefore, $Max_{Processlevel,1}=12$.
2. Annotation Consistency on element *e* for process *p,* $AC_{e,p} = (Max_{e,p} * 100)/N$, where N is the total number of participants. For the example given above, $AC_{Processlevel,1} = (12 * 100)/20 = 60$.

3. The average AC for an element *e*, $AC_e = (\sum AC_{e,p}) / n$ for $p = 1 \ldots n$, where n is the number of annotated processes.

The existence of similarities in process annotation means that there is a common understanding of the BPMM between different people. This implies that the process metadata based on BPMM will communicate the same meaning to different people.

Variable 2. *Annotation Correctness (AR)*: It is the degree to which process annotation (using BPMM) by different people is correct. AR is measured by the number (in percentage) of participants who correctly annotated a process for an element of BPMM.

The majority of participants may have a common but incorrect understanding of the BPMM model. Therefore, in order to determine whether the process annotation by participants is correct or not, the AR is measured.

For measuring the AR, the process annotation from different participants is compared with the process annotation from the inventors of BPMM, assuming that the inventors' annotation is correct. AR is computed as follows:

1. Let $C_{e,p}$ be the number of participants with correct (identical to inventors') annotation on element *e* for process *p*. Where, *e* is an element in {*Resource*, *Actor*, *Process Level*, *Process Relationship*, *Process Area*, *Process Phase*, *Process Type*}.

   For example, suppose a process (*p=1*) is annotated by 20 participants and for an element (*Process Level*), out of 20 participants 12 annotate it as 'operational', 5 as 'tactical', and 3 as 'strategic'. Where, the correct (inventors') annotation is 'tactical'. Therefore, $C_{Processlevel,1}=5$
2. The Annotation Correctness on element *e* for process *p* $AR_{e,p} = (C_{e,p} * 100)/N$, where N is total the number of participants. For the example given above $AR_{Processlevel,1} = (5 * 100)/20 = 25$. Similarly, if the correct (inventors') annotation is 'operational' then $AR_{Processlevel,1} = 60$.
3. The average AR for an element *e*, $AR_e = (\sum AR_{e,p}) / n$ for $p = 1 \ldots n$, where n is the number of annotated processes.

The existence of similarities in process annotation (between participants and inventors) means that the BPMM model elements are correctly understood. This implies that, the process metadata produced will be free of errors.

Variable 3. *Perceived Ease of Use (PEOU)*: The degree to which a person believes that using the BPMM model for annotating processes would be free of effort. In order to investigate perceived ease of use we asked the participants to assess two statements on a scale of 1 to 5 (Strongly Disagree to Strongly Agree). The statements are, PEOU1 (The annotation definitions are clear and helpful for annotation), PEOU2 (It was easy to annotate the business processes).

3.5 Results and Discussion

In this section, the data collected from the experiment are analyzed and discussed in order to evaluate the BPMM elements definitions. For the analysis, the mean and the standard deviation of annotation consistency and correctness for each element are computed. Tables 1 and 2 and fig. 3 show the summary of statistics of process annotation.

Table 1. Annotation Consistency

	Resource	Actor	Process Level	Process Relationship	Process Area	Process Phase	Process Type
Mean	86.56	76.46	69.58	80.56	44.62	48.69	62.24
StDIV	12.40	12.24	21.19	9.62	3.23	14.16	14.60

Annotation Consistency (AC). The data in table 1 show that more than 62% of participants have identical process annotation for the following elements *Resource, Actor, Process Level, Process Relationship,* and *Process Type*. This indicates that there is a common understanding of these BPMM elements between different users. However, the *Process Area* and *Process Phase* elements have less than 50% of participants with identical process annotation. This indicates that the *Process Area* and *Process Phase* definitions are differently understood by the participants.

Table 2. Annotation Correctness

	Resource	Actor	Process Level	Process Relationship	Process Area	Process Phase	Process Type
Mean	86.56	76.46	69.58	80.56	36.84	42.64	54.66
StDIV	12.40	12.24	21.19	9.62	9.21	17.86	20.06

The data also show that the annotation consistency (similarity) of the same element varies between business processes. This is shown by high standard deviation i.e. 21.19 for the *Process Level* element.

Annotation Correctness (AR). The data in table 2 show that more than 54% of the participants have correctly annotated business processes for the elements *Resource, Actor, Process Level, Process Relationship,* and *Process Type*. This indicates that the BPMM elements definitions are well understood by different people, implying that most process metadata generated by users based on the model will be free of errors. However, less than 50% of participants have correct process annotation for the elements *Process Area* and *Process Phase*. The detailed analysis shows that participants who correctly annotated *Process Area* and *Process Phase* had more industrial experience compared to others. While the two elements are based on widely accepted frameworks [17, 21, 22], understanding and applying these definitions to annotate business processes seems to require some basic industrial experience, which many participants lacked.

Comparing the annotation consistency (AC) and correctness (AR), fig. 3 shows that AC is equal to AR for the elements *Resource, Actor, Process Level and Process Relationship.* This means that, for these elements, the majority of the participants who identically annotated the processes were correct. Therefore, the majority of the participants have a common and correct understanding of the BPMM and its element definitions. However, the definitions of Process Areas, Process Phase and Process Type were not correctly annotated so we hypothesize that these definitions need to be sharpened.

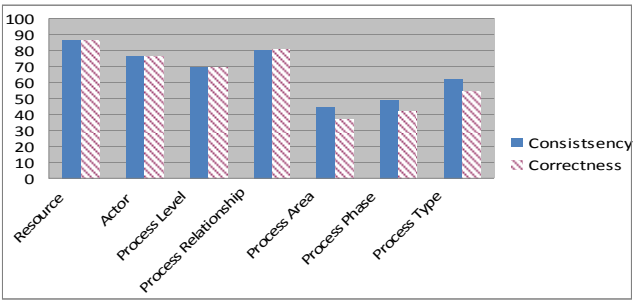


Fig. 3. Annotation Consistency and Correctness

Perceived Ease of Use (PEOU). Fig. 4 shows the summary of statistics for user perception of the model. More than 52% of the participants agree (agree and strongly agree) with PEOU1 (*The annotation definitions are clear and helpful for annotation).* Whereas more than 58% of the participants agree (agree and strongly agree) with PEOU2 (*It was easy to annotate the business processes*).

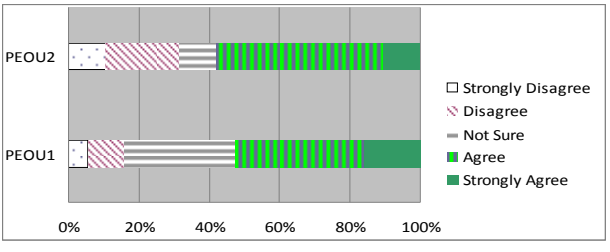


Fig. 4. Perceived Ease-of-Use

4 Conclusion

In this paper we have proposed a Business Process Metadata Model (BPMM) that can be used for annotating business processes to facilitate searching, interpreting process models as well as navigating the repository. BPMM is composed of elements derived from concepts which were elicited (from literature) and validated through an empirical study. While the complete validation of the model can only be achieved after implementation, the model is empirically evaluated through a controlled experiment to measure consistency and correctness of process annotation.

From the study, we have learned that the annotations of most of the BPMM elements by different people are identical and correct. This implies that the given definitions of BPMM elements are understandable. However, specifically the definitions for Process Area, Process Phase and Process Type need to be sharpened in order to increase annotation consistency and correctness.

One of the limitations of the study is that the annotation consistency and correctness for two elements (business goals and business context) is not measured. This is

due to the reason that there are no widely accepted values for the two elements. Future research aims at further validating the BPMM.

References

1. Markovic, I., Pereira., A.C.: Towards a Formal Framework for Reuse in Business Process Modeling. In: Proceedings of the 2nd International Workshop on Advances in Semantics for Web Services, in conjunction with BPM 2007, Brisbane, Australia, pp. 484–495 (2008)
2. Hornung, T., Koschmider, A., Oberweis, A.: A Recommender System for Business Process Models. In: Proceedings of the 17th Annual Workshop on Information Technology and Systems (WITS 2007), Montreal, Canada, pp. 127–132 (2007)
3. Rodrigues, J.A., Souza, J.M., Zimbrao, G., Xexeo, G., Neves, E., Pinheiro, W.A.: A P2P Approach for Business Process Modelling and Reuse. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 297–307. Springer, Heidelberg (2006)
4. Ma, Z., Leymann, F.: A Lifecycle Model for Using Process Fragment in Business Process Modeling. In: Proceedings of the 9th International Workshop on Business Process Modeling, Development and Support, in conjunction with CAiSE 2008, France (2008)
5. Yan, Z., Dijkman, R., Grefen, P.: Business Process Model Repositories - Framework and Survey. Beta Working Papers, vol. 292. Eindhoven University of Technology (2009)
6. Lusk, S.: The Value of a Formal Business Process Repository (2007), `http://www.bpminstitute.org/articles/article/article/the-value-of-a-formal-business-process-repository.html`
7. Vanhatalo, J., Koehler, J., Leymann, F.: Repository for Business Processes and Arbitrary Associated Metadata. In: BPM Demo Session of the 4th International Conference on Business Process Management, CEUR Proceedings, Austria, vol. 203, pp. 25–31 (2006)
8. Broekstra, J., Ehrig, M., Haase, P., van Harmelen, F., Kampman, A., Sabou, M., Siebes, R., Staab, S., Stuckenschmidt, H., Tempich, C.: A metadata model for semantics-based peer-to-peer systems. In: Proceedings of the 1st Workshop on Semantics in Peer-to-Peer and Grid Computing, in conjunction 12th WWW Conference, Hungary (2003)
9. Gao, S., Krogstie, J.: Facilitating Business Process Development via a Process Characterizing Model. In: International Symposium on Knowledge Acquisition and Modeling 2008. IEEE CS, Los Alamitos (2008)
10. The UN/CEFACT: Common Business Process Catalog (CBPC). Technical Specification, Version 1.0 (2005), `http://www.uncefactforum.org/TBG/TBG14/TBG14Documents/cbpc-technical-specification-v1_0-300905-11.pdf` (last accessed Feb ruary 18, 2010)
11. MIT Process Handbook, `http://process.mit.edu/Directory.asp?ID=114&Expand=92` (last accessed February 18, 2010)
12. UN/CEFACT: Context and Re-Usability of Core Components, Version 1.04 (2001), `http://www.ebxml.org/specs/ebCNTXT.pdf` (last accessed on February 18, 2010)
13. Kueng, P., Kawalek, P.: Goal-based Business Process Models: Creation and Evaluation. Business Process Management Journal 3(1), 17–38 (1997)
14. Lin, Y., Solvberg, A.: Goal Annotation of Process Models for Semantic Enrichment of Process Knowledge. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 355–369. Springer, Heidelberg (2007)

15. Soffer, P., Wand, Y.: On the Notion of Soft Goals in Business Process Modeling. Business Process Management Journal (2005)
16. UN/CEFACT Common Business Process Catalog (CBPC) Team. White paper, Reference Model for Web services Classification in Global Registry
17. McCarthy, W.E.: The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. The Accounting Review (1982)
18. Dunn., C., Cherrington, O.J., Hollander., A.: Enterprise Information Systems: A Pattern-Based Approach. McGraw-Hill, Irwin (2004)
19. Curtis, B., Kellner, M.I., Over, J.: Process Modeling. Communications of ACM 35(9), 75–90 (1992)
20. http://poem.dsv.su.se/ (last accessed February 17, 2010)
21. Porter, M.: Competitive Advantage. Free Press, New York (1985)
22. OPEN-EDI Phases, http://jtc1sc32.org/doc/N1301-1350/32N1337T-FCD15944-4.Doc (last accessed February 18, 2010)
23. Anthony, R.N.: Planning and Control Systems: A Framework for Analysis. Harvard Business School, Cambridge (1995)
24. Anthony, R.N., Dearden, J., Bedford, N.M.: Management Control Systems. Irwin Homewood, IL (1995)
25. SCOR Reference Model, http://www.supply-chain.org/ (last accessed Febuary 18, 2010)
26. Hofreiter, B., Huemer, C.: From a UMM Business Process Model to a Business Environment Specific ebXML Process. Journal of E-Commerce Research, JECR (2006)
27. Rosemann, M., Recker, J., Flender, C.: Contextualisation of business processes. IJBPIM 3(1), 47–60 (2008)

Exploring Intuitive Modelling Behaviour

Ilona Wilmont[1], Sjaak Brinkkemper[2], Inge van de Weerd[2], and Stijn Hoppenbrouwers[1]

[1] Institute for Computing and Information Sciences, Radboud University Nijmegen, Nijmegen, The Netherlands
i.wilmont@science.ru.nl, stijnh@cs.ru.nl

[2] Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands
{s.brinkkemper,i.vandeweerd}@cs.uu.nl

Abstract. Understanding modelling behaviour is an important step towards situated modelling support, especially when aiming to actively involve the domain expert in modelling without expert interventions. In search for a hypothesis on which modelling acts humans exhibit naturally, this paper presents an exploratory study into the modelling approaches intuitively taken by people trained in modelling as opposed to people not trained in modelling. Participants were asked to create a concept map of either a familiar or unfamiliar knowledge domain.

Analysis shows that there are differences between the approaches novice and expert modellers follow, the decisions they make in representing an aspect or not, and the level of abstraction they choose.

Keywords: modelling behaviour, situated modelling support, intuitive decision making, abstraction.

1 Towards Situated Modelling Support

Conceptual modelling is a popular, widely used technique in the IT industry [2]. Process modelling is the main motivation to engage in modelling, and the use of official modelling techniques increases significantly in large organisations, due to the increasing complexity of projects [2].

In this paper we consider the following question: *What thought processes do novice modellers intuitively unveil when modelling without specific restrictions, and how does this compare to how modelling experts work and think?* There are two main assumptions on which modelling approach is most intuitive for people to follow: thinking in processes, or thinking in terms of objects essential to the domain. Indeed the use of both approaches by different information engineers has been documented [28]. We propose that this distinction may not be so strict. Intuitive modelling as displayed by novices shows that object-driven and process-driven thoughts occur in parallel, each object triggering a related process and vice versa.

The importance of being able to teach and guide novices in making well-structured, unambiguous and syntactically correct models is emphasised by the

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 301–313, 2010.

fact that domain experts, finding themselves placed in the role of novice modellers, view modelling as a complicated process [9,11]. After a single, experiential modelling training they are often expected to participate actively in modelling sessions. While experiential training is more effective than didactical training or no training when it comes to imparting modelling knowledge [27], modelling languages have clearly not been designed with the novice in mind. The quality of the models and the shared understanding the novices should have of it are often negatively affected by this.

Teaching modelling should therefore be done in a way that appeals to the novices' way of thinking, is easy to understand and learn for them and ultimately is also fun to work with. While there is an extensive body of research available on specific modelling languages [21,10,8,25], the above mentioned *human* aspects of modelling have received relatively little attention so far.

First of all, we describe the hypothesis and the experimental setup. Then, we explain how the code system for analysis was developed and applied, followed by an overview of the most interesting observations. Finally, a brief discussion of abstraction and information modelling as interpreted in this paper is given.

2 The Experimental Setup

The empirical work described below is of an exploratory nature and meant solely for the purpose of hypothesis generation. The authors acknowledge that the number of participants used is too small to draw any significant conclusions. Therefore, it should be noted that this paper presents work in progress. The hypothesis aims to provide a basis for understanding modelling behaviour.

2.1 Hypothesis

Conceptual modelling centres around two main processes: conceptualisation and constructing relations between essential concepts. Modellers tend to begin with several concepts they consider essential to the domain, and from there a process-oriented stream of thought is triggered relating the concepts to one another. Thoughts are supported by a mental walkthrough, relating the abstract domain to a specific instance, or instances, of it well-known to the modeller. Especially in novices, mental walkthroughs are visual.

2.2 Capturing Human Thought

To begin with, we explain the formalism we used to allow the participants to represent their thoughts in the models.

Mental Models. Mental models are a fundamental aid to describing how an individual stores knowledge. Rouse and Morris [24] provide a functional definition that very well suits the purposes of this research:

> "Mental models are the mechanisms whereby humans are able to generate descriptions of system purpose and form, explanations of system functioning and observed system states, and predictions of future system states. " [24]

Mental models are therefore cognitive representations, unique to the individual, based on the individual's experiences and expectations, to guide behaviour, organise thoughts and influence the interpretation of information. They tend to be dynamic rather than static, and can be manipulated by the individual to predict the outcome of a certain problem-solving strategy [27,7]. The accuracy of mental models often influences the quality of problem solutions [27]. The reason for that may be that people base their solutions on their mental models of a problem and a situation, and if those mental models happen to be inaccurate or incorrect, the solution will inevitably suffer too [14,27]. Research indicates that in collaborative situations, teams who have shared mental models perform better than those who do not [27].

Concept Maps. Mental models can be visually represented using the technique of *concept mapping* [20,7,26]. Concept maps are basic models representing concepts and the associations between them, using circles for concepts and directed arrows for associations. Gwee [7] has used concept mapping for the measurement of mental models, arguing that they are "easy to administer and participant friendly". Indeed they require little cognitive effort from the participant, thereby minimising compromise of result validity [7].

For these reasons we choose to employ concept mapping as the main formalism in our study.

2.3 Participants

The distinction between *expert* and *novice modellers* is made, mimicking the difference between information engineers and domain experts in real situations. Expert modellers are people from the IT industry who include some form of information systems modelling in their daily tasks, whereas novice modellers are people who have no experience in information systems modelling.

The study has a between-groups and within-groups setup. A total of 10 participants was used, randomly split into two groups of 5 participants. Both groups consisted of 2 expert modellers and 3 novice modellers. The participants came from diverse backgrounds. Experts were recruited from industry and university (students already in a job). Novice modellers were recruited from any industry not involving IT as their main service. The minimum age limit was 18, because abstraction capacity is thought to start developing when children enter adolescence [23], and is likely to be well-developed by the time they reach adulthood. Therefore, participants should be fully capable of carrying out the abstraction steps required for modelling.

2.4 Procedure

The procedure involved two different tasks, one for each group. While carrying out the task, the participant was asked to think aloud.

Task 1. Participants were asked to create a conceptual domain model of a library. Participants were told to use only blocks and directed arrows (the syntax of a concept map), although they were free to expand this basic syntax if they felt it was necessary. There was no time limit, and the stopping criterion was that all relevant aspects, in the participant's perception of a library, should be in the model. After that they were asked about their thought processes, why they chose the extended representation they used and whether they found modelling intuitive.

Task 2. Participants were first asked to view a 30-minute fragment from the BBC documentary 'Journey of Life', episode 1 - 'Seas of life' about the first phases of evolution. Participants had a basic knowledge of evolution. During the film, participants were allowed to take notes in whatever form they liked, such as plain text, structured text, schemes etc. After the film, the participants were asked to create a concept map (with the possibility of extended syntax) of what they had just seen: what had happened during the first phases of evolution. The rules were the same as in the other task, and the participants were also interviewed afterwards.

3 The Code System and Analysis

The data was analysed by using a code system to add structure to the data. In this exploratory phase, we forego any attempt at statistical analysis of code usage, because we believe it is important first to get a qualitative description of the observed phases of modelling behaviour.

All experiments were recorded on video and transcribed. The transcripts were analysed using a code system derived from literature. Four code families were created.

The family *Type of relation between concepts in mind* is based on [7]. According to Gwee, stored knowledge in humans consists at the most fundamental level of concepts associated in the individual's mind, connected with the following possible relations: association (A related to B), similarity (A similar to B), hierarchical (B subset of A), causal (A causes B) and contiguity (B follows A) [7]. By marking how often these relations occur in modelling acts by novices and experts, we can get an idea of which relations are most natural to the human mind.

The code family *Phases in process of modelling* was formed by combining a view from modelling research [22] with a scheme of human information processing from psychology [15]. The code 'Abstraction' appeared prominently in both schemes, and therefore our interpretation of abstraction receives further attention in section 5.2.

The third family *Metaphor of human thinking* is based on the literature by [5,6,12] and further accounts for the psychological processes performed by the individual. Though all metaphors, discussed in more detail in section 5.1, represent relevant characteristics of thought, only *Habit* and *Awareness* have proved

themselves useful to be included in the coding system for analysis. Habit points out how previous habits influence modelling behaviour, in which cases habits are useful and in which cases habits may present obstructions to easy modelling. Awareness and related shifts in awareness allow us to track which aspects receive most attention and how people browse through the domain in their minds.

An overview of the codes sorted by code family is given in table 1.

Table 1. An overview of the codes sorted by code family

Type of relation between concepts in mind	**Phases in process of modelling**	**Metaphor of human thinking**
Association	Abstraction	Habit
Causal	Generalisation	Stream of thought
Contiguity	Selection	Awareness
Hierarchical	Classification/ Categorisation	Incomplete utterances
Similarity	Storage	Knowledge under construction
	Structuring	
	Reflection	

Finally, a fourth code family, *Modelling concepts used* was created. This code family is not based on literature, but rather on observations of which concepts the participants used in their models. There are some concepts used by all modellers, which implies they seem to be in harmony with an individual's natural thought patterns. Other concepts were only seen in models created by experts, which suggests these have been learned and based on habits from everyday modelling languages used and tasks done. Apart from the codes, a memo was attached to each utterance in the transcripts describing exactly what happens with regard to the observable process of modelling behaviour.

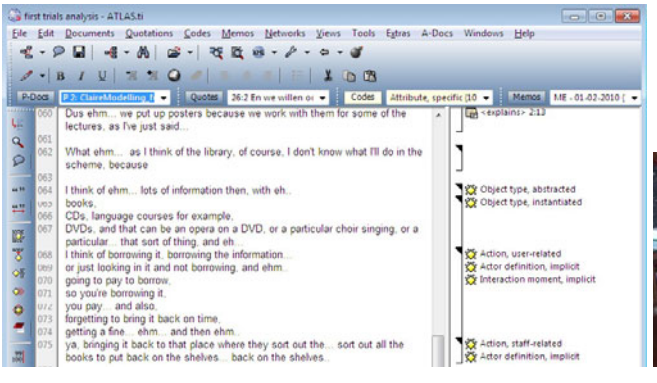
The codes linked to the transcripts give an indication of which concepts are most often used, which is crucial as an impression of intuitive modelling concept use. An overview of the codes with their number of occurrences is given in table 2.

4 Results

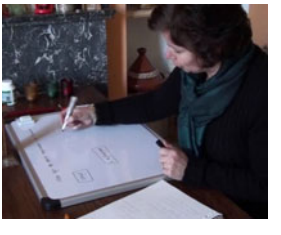
From the observed modelling behaviour and the modelling concepts used, it can be said that both novices and experts alike are comfortable with thinking about a domain from a user oriented perspective. Also both novices and experts include staff in the model, but thinking in terms of organisational processes is only shown by experts, implying that this way of thinking is acquired as a specialised skill. Figure 2 shows a schematic overview of all specific concepts used in the models. Light grey concepts are used by both novices and experts, white concepts are only used by experts, and dark grey concepts are only used by novices.

Table 2. Overview of the specific modelling concepts used

Modelling concepts used	**Number of occurrences**
Action, staff-related	7
Action, user-related	11
Actor definition, explicit	4
Actor definition, implicit	9
Attribute, generic	4
Attribute, specific	10
Relation, implicit	1
Interaction moment, explicit	6
Interaction moment, implicit	4
Object type, abstracted	2
Object type, instantiated	7
Expert use only	
Relation, explicit	8
Business type	2
Constraint/condition	2
Data type assigned	4



(a) An example of codes attached to transcripts



(b) Novice modeller at work

Fig. 1. Examples of data and data analysis

However, this does not imply that thinking in processes in the more general sense is a specialised skill. The general way of thinking observed in all participants was the following: first an *ad hoc* mental walkthrough of the domain was done, resulting in a set of concepts capturing the essence of the domain. These concepts were all semantically related and therefore thinking about one triggered recall of the other. After the set of essential concepts had been written down, a phase of process-oriented thinking was triggered, based on functional questions of what one can do in the domain or what is happening in it.

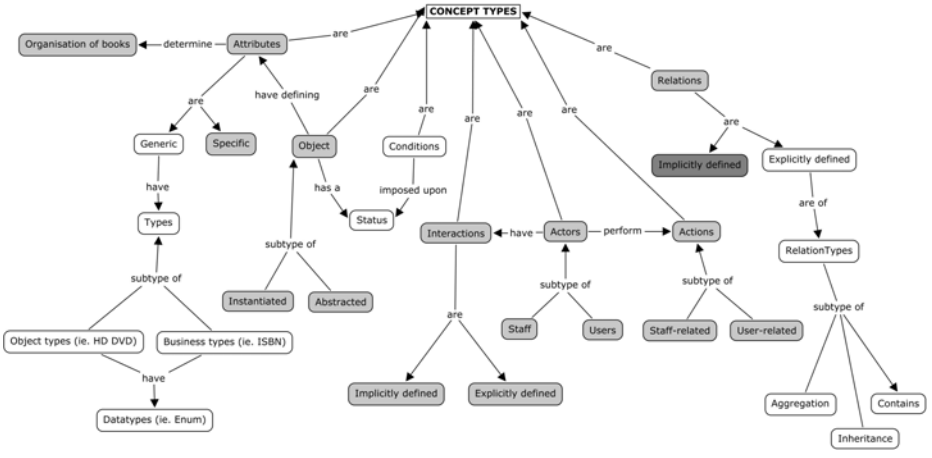


Fig. 2. An overview of the specific modelling concepts used by the participants

Novice participants mentioned that they used visual cues in their minds, either of a specific library or fragments from the film, to help them formulate concepts about and support their thoughts on what was happening. They performed and experienced certain processes in their minds and wrote them down in the model exactly as such. A certain form of structuring, crude syntax differentiation and abstraction was taking place, but there was no consistency in the level of abstraction or syntax structure used. Novices felt the need for this structuring but could not by themselves be consistent enough.

Before they started modelling, novices needed some time to verbalise their thoughts to create structure in the possibilities. This phase of generating knowledge was highly explicit in novices. Moreover, they were incapable of making a proper selection of the concepts essential for a conceptual model. They simply put in everything they thought of. During the film fragment trials, this effect was less obvious since the film fragment had already done some selecting for them.

In contrast, even though the experts also mentioned making a link to the concrete situation, the way in which this was done was immediately linked to an abstract categorisation of the concepts. There was strict syntax differentiation and re-use of previously defined relations, as well as consistency in abstraction and structure. Experts mentioned doing a mental walkthrough of the different processes in the domain on a functional level: placing themselves in the position of the organisation. This attitude marks a key difference between experts and novices and is probably acquired by instruction but even more by experience.

Finally, experts were able to do immediate reflection on their model as they were creating it. When things did not fit in satisfactorily, they were easily altered, whereas novices only felt doubt about their representation but could not derive any feedback from the discrepancies.

In figure 3 a colour-coded schematic overview is given of all thought processes observed. Generally speaking, the observations suggest that there is a certain key

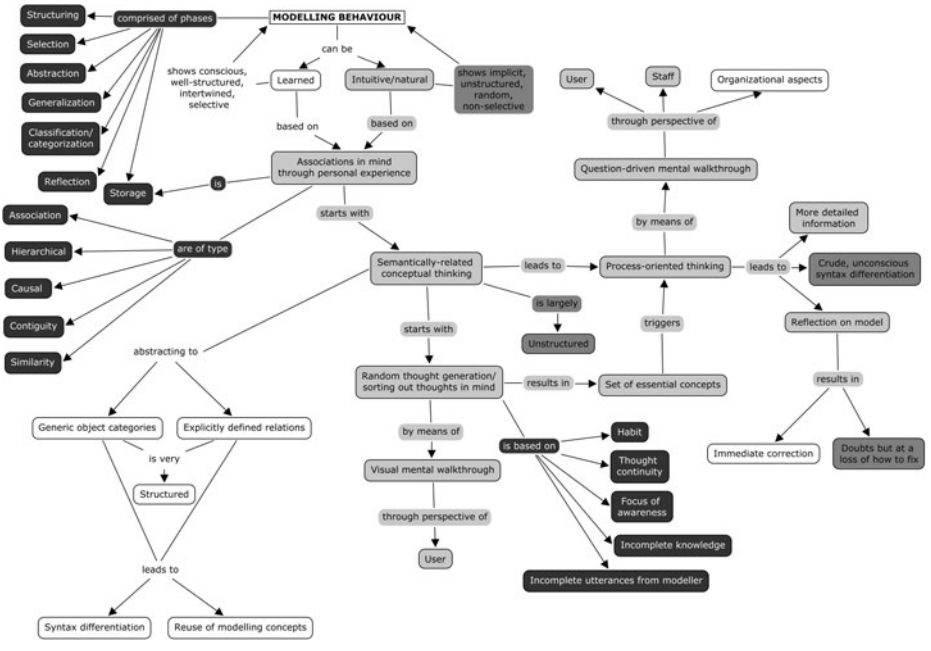


Fig. 3. An overview of the thought processes used by the participants

thought procedure that both experts and novices follow during modelling (light grey route). However, there are certain processes only performed by experts (white route) which they manage to integrate seamlessly in the entire key procedure. The dark grey route refers to phenomena only observed in novices, and the black route represents the code families in relation to the model of modelling behaviour.

5 The Human Aspects of Modelling

In this section we further elaborate on the theory of thought used in the creation of the code system. Also, a brief discussion is necessary on the background of conceptual models and modelling as interpreted in this paper. Since the focus is on exploring conceptualisation in and by the modeller, no specific distinction is made between what kind of models are created, thereby justifying the use of the generic term ‘conceptual models’.

5.1 Characteristics of Thought

In [5,19], five metaphors of human thinking are discussed, based on [13]. These metaphors have been successfully incorporated in usability research. Using evaluation guidelines based on the thought metaphors below has resulted in “finding more problems ... of a more complex nature and ... more likely to persist for expert users” [6], as compared to conventional usability measurement methods.

1. **Habit**
   Habit plays an important role in human thought and physical actions, such as automaticity, all linguistic activity, habits of reasoning.
2. **Stream of thought**
   Humans experience their thoughts as a continuous, dynamic stream, representing the full "richness and wholeness of a person's mental objects".
3. **Awareness**
   Awareness is shaped "through a focus of attention, the fringes of mental objects, association, and reasoning".
4. **Incomplete utterances**
   "The incompleteness of utterances in relation to the thinking underlying them and the ephemeral nature of those utterances".
5. **Knowledge under construction**
   The dynamics of human knowledge, always changing and incomplete.

5.2 Abstraction

When modelling, one purposely abstracts from observable facts in an attempt to create a generic description of a complex situation, yet the link to the real situation should not be lost. The ability to take an abstract point of view varies per individual, and there is likely to be a clearly observable difference between the ways in which experts and novices employ abstraction during modelling.

There are many ways to define abstraction, depending on which perspective is taken. In fields such as philosophy, mathematics and logic, abstraction is characterised as *information neglect*: "eliminating specificity by ignoring certain features" [1]. Whereas the rigid nature of abstractions in mathematics allows ignoring of information, the highly dynamic and interactive nature of computer science is fundamentally different and therefore requires a different interpretation: *information hiding* [1]. A key concept in information hiding is the *deliberate omission of irrelevant information so that the focus is only on the relevant aspects of conceptualisation.* A simple utterance such as 'The desk is brown' is already a form of abstraction, since many details are left out, such as the material it is made of, its exact size, its shape etc. Yet the assumption is that those details still exist, and may at some point become relevant again.

In this paper the perspective of information hiding is used when talking about abstraction. We have no objective way of measuring abstractions during modelling, therefore modelling acts employing abstraction are identified subjectively by the researchers.

5.3 Models

Some words need to be spent on the definition of a model. Generic dictionary definitions emphasise the notions of *representation* and *visualisation* as being central to a model. A more specific, IT-oriented definition is the one formulated by Dietz [3]: *"Any subject using a system A that is neither directly nor indirectly interacting with system B to obtain information about the system B, is using A*

as a model for B." Dietz explains that A can (but need not necessarily) be a conceptual, abstracted representation of B, bearing some sort of similarity to the system B such that it can take on the *role* of a model.

For the purposes of this research, we provide the following definition of a model, centred around the key concepts discussed above:

> *A purposely abstracted, visual or textual representation of a clearly demarcated part of what the modeller perceives as reality, or of a situation, not necessarily in existence yet, which the modeller perceives to be efficient and logical.*

5.4 Modellers

A realistic modelling situation typically involves both stakeholders, or domain experts, and expert modellers, or system analysts. While it is clear that both parties lack certain knowledge and are therefore required to complement one another, this is not always a self-evident process. Problems in knowledge transfer can occur due to difficulties in communication or understanding, use of jargon, limited time or lack of empathy [11]. Apart from this, there are basic skills which are required of all participants [4]. The following is a short overview of the most important skills needed to come to a model (based on [4]).

Domain Experts. Domain experts must be able to provide complete information. While this may seem evident, in practice it can be extremely difficult to come up with every single relevant aspect of the application domain. Domain experts must be able to organise the information in accordance with the dynamics of their application domains, to validate any description of the domains and to judge the significance of each part of the description. Another desirable skill is that domain experts can think on an abstract level.

System Analysts. System analysts are required to handle implicit knowledge, which is possessed by the domain expert. System analysts must be able to detect the presence of this knowledge and elicit it in an appropriate way. They must be able to validate a description of the domain for consistency and match the domain expert's utterances to concepts of the modelling technique in use. The ability to think abstractly is even more important for system analysts, and last but not least they must be able to understand the domain expert's world view. This is very important when it comes to eliciting tacit knowledge.

5.5 Modelling

The research presented is an exploratory study of behaviour shown by people involved in a conceptual modelling process. For a detailed discussion of conceptual modelling, see [4,28,29]. In [28,29], a study on the way of working as applied by information engineers is described. Conceptual modelling is an iterative process, leading to more structured modelling tasks and models as the process advances.

Also, more refined modelling concepts are used as the model becomes more detailed. No strict order of performing tasks is adhered to. Modelling tasks are performed when there is a need to gain insight into a certain part of the problem area, or to communicate aspects of the problem area to users. It is very much dependent on the nature of the problem and the domain. The part of the problem domain which is paid attention to is also determined by need. Verhoef [28] found data models and process models to be used interchangeably, even though the use of different strategies for data modelling and process modelling were observed.

6 Discussion and Conclusions

Efforts to improve usability of modelling techniques have so far mainly focused on making the syntax appear more intuitive or self-explanatory to the user, for instance by colouring or shape consistency [18,16,17]. In this study, we have tried to assess what type of thought comes naturally to novice modellers, so that we may eventually design modelling support that supports the novice's way of thinking.

One of the main obstacles in this study is the small sample size. However, it was interesting to observe that all participants followed the same basic, high-level thought process of trying to grasp the essence of the domain before proceeding to a process-oriented thought pattern to recall more of the functionality within the domain.

The main differences between experts and novices are that experts have a much richer mental model of available modelling concepts and that they integrate all phases of modelling into their behaviour, such as recalling facts, abstracting, structuring, generalising, selecting and reflecting, and are able to use each function immediately whenever they feel the need.

Novices show much clearer demarcation between the phases, and also use fewer phases. They clearly start with the recall phase, generating whatever they know about the domain. Then, crude abstractions are made, structuring knowledge as it is stored in their minds, and seemingly omitting the phases of generalising, selecting and reflecting. Yet these phases are crucial when it comes to consistently including relevant information at the right level of abstraction in the model.

Observation of experts does suggest that a modelling mindset utilising strict consistency in categorisation and syntax use is very much a specialised skill, but above all, that such a mindset comes from a great deal of everyday experience, since the industry experts were a lot stricter with this than the student expert.

In conclusion, it may therefore be said that support for the novice while performing the desired skills [4] should focus on aiding consistency, finding the right level of abstraction, selecting the relevant concepts and supporting on-the-fly reflection on the model. This support might take the form of guidance questions, either by means of an automated tool or provided by a facilitator. The best way for providing support still remains an issue for future research, and may also depend on the modeller in question.

Further research will focus on verifying the proposed hypothesis with a larger sample size. A stronger link to psychological theory of human thinking is important, as is a further extension of individual thought processes to behaviour in collaborative modelling situations, since all participants said that they would rather have carried out the modelling task with two or three people. Seeing how individual behaviour merges into group behaviour will be an interesting aspect to consider.

References

1. Colburn, T., Shute, G.: Abstraction in computer science. Minds and Machines 17(2), 169–184 (2007)
2. Davies, I., Green, P., Rosemann, M., Indulska, M., Gallo, S.: How do practitioners use conceptual modeling in practice? Data & Knowledge Engineering 58(3), 358–380 (2006)
3. Dietz, J.L.G.: Enterprise ontology: theory and methodology. Springer, Heidelberg (2006)
4. Frederiks, P.J.M., van der Weide, T.P.: Information modeling: the process and the required competencies of its participants. Data & Knowledge Engineering 58(1), 4–20 (2006)
5. Frøkjær, E., Hornbæk, K.: Metaphors of human thinking in HCI: Habit, stream of thought, awareness, utterance, and knowing. In: Proceedings of HF2002/OzCHI 2002, pp. 25–27. Basic Books, New York (2002)
6. Frøkjær, E., Hornbæk, K.: Metaphors of human thinking for usability inspection and design. ACM Transactions of Computer-Human Interactions 14(4) (2008)
7. Gwee, K.: Measuring mental models. In: IMTA Annual Conference 2005 (2005)
8. Halpin, T.: Object-role modeling (ORM/NIAM). In: Handbook on Architectures of Information Systems, pp. 81–101 (1998)
9. Hoppenbrouwers, S.J.B.A.: Community-based ict development as a multi-player game. In: Benoit-Barn, C., Brummans, B.H., Cooren, F., Giroux, H., Létourneau, A., Raymond, D., Robichaud, D. (eds.) What is an organization? Materiality, agency, and discourse: a tribute to the work of James R. Taylor, Dept. of Organizational Communication, University of Montreal (May 2008)
10. Hoppenbrouwers, S.J.B.A., van Bommel, P., Jarvinen, A.: Method Engineering as Game Design: an Emerging HCI Perspective on Methods and CASE Tools. In: Workshop proceedings of EMMSAD 2008: Exploring Modeling Methods for Systems Analysis and Design, affiliated to CAiSE 2008 (2008)
11. Hoppenbrouwers, S.J.B.A., Weigand, H., Rouwette, E.A.J.A.: Setting rules of play for collaborative modelling. International Journal of e-Collaboration, Special Issue on Collaborative Business Information System Development 5(4), 37–52 (2009)
12. Hornbæk, K., Frøkjær, E.: Evaluating user interfaces with metaphors of human thinking. In: Carbonell, N., Stephanidis, C. (eds.) UI4ALL 2002. LNCS, vol. 2615, pp. 486–507. Springer, Heidelberg (2003)
13. James, W.: The Principles of Psychology. Harvard University Press, Cambridge (1983)
14. Kempton, W.: Two theories used of home heat control. Cognitive Science 10, 75–91 (1986)
15. McCabe, V., Balzano, G.J.: Event cognition: an ecological perspective. Lawrence Erlbaum Associates, Mahwah (1986)

16. Mendling, J., Recker, J., Reijers, H.A.: On the usage of labels and icons in business process modeling. International Journal of Information System Modeling and Design, 2009a(forthcoming) (2009)
17. Mendling, J., Reijers, H.A., Recker, J.: Activity labeling in process modeling: Empirical insights and recommendations. Information Systems, IS (2009)
18. Moody, D.L., Heymans, P., Matulevicius, R.: Improving the Effectiveness of Visual Representations in Requirements Engineering: An Evaluation of i* Visual Syntax. In: 2009 17th IEEE International Requirements Engineering Conference, pp. 171–180. IEEE, Los Alamitos (2009)
19. Naur, P.: Chi and human thinking. In: Proceedings of the 1st Nordic Conference on Computer-Human Interaction (NordiCHI 2000), Stockholm, Sweden, October 23 - 25. ACM, New York (2000)
20. Novak, J.D., Cañas, A.J.: The theory underlying concept maps and how to construct and use them. Technical report, Florida Institute for Human and Machine Cognition (2008)
21. Oei, J.H.L., van Hemmen, L.J.G.T., Falkenberg, E.D., Brinkkemper, S.: The meta-model hierarchy, a framework for information systems concepts and techniques (1992)
22. Persson, A.: Keynote speech. In: 4th SIKS/BENAIS Conference on Enterprise Information Systems (2009)
23. Piaget, J.: Zes psychologische studies. Van Loghum Slaterus (1969)
24. Rouse, W.B., Morris, N.M.: On looking into the black box: prospects and limits in the search for mental models. Psychological Bulletin 100(3), 349–363 (1986)
25. Rumbaugh, J., Jacobson, I., Booch, G.: The Unified Modeling Language Reference Manual. Pearson Higher Education (2004)
26. Siau, K., Tan, X.: Improving the quality of conceptual modeling using cognitive mapping techniques. Data & Knowledge Engineering 55(3), 343–365 (2005)
27. Van Boven, L., Thompson, L.: A look into the mind of the negotiator: Mental models in negotiation. Group Processes & Intergroup Relations 6(4), 387 (2003)
28. Verhoef, T.F.: Effective Information Modelling Support. PhD thesis, Delft University of Technology (1993)
29. Verhoef, T.F., Ter Hofstede, A.H.M.: Feasibility of flexible information modelling support. In: Iivari, J., Rossi, M., Lyytinen, K. (eds.) CAiSE 1995. LNCS, vol. 932, pp. 168–185. Springer, Heidelberg (1995)

Co-evolution of (Information) System Models

Ajantha Dahanayake[1] and Bernhard Thalheim[2]

[1] Georgia College and State University, J. Whitney Bunting School of Business
Dept. of Information Technology and Marketing, Campus Box 12,
Milledgeville 31061, GA, USA
ajantha.dahanayake@gcsu.edu
http://hercules.gcsu.edu/~adahanay/
[2] Christian Albrechts University Kiel, Department of Computer Science, Olshausenstr. 40,
D-24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de
http://www.is.informatik.uni-kiel.de/~thalheim

Abstract. Information systems' modelling is based on separation of concern such as separation into facets or viewpoints on the application domain from one side and separation of aspects (structuring, functionality, interactivity, distribution, architectural components) from the other side. Facets and aspects are typically specified through different models that must be harmonised and made coherent. Such varieties of models are difficult to handle, to evolve, to maintain and to use. Most design methodologies adopt the master-slave principle in order to handle the coherence of such model assemblies by assigning one model to be the master and mapping the master to slave models. Moreover, these models diagrams are typically not developed from scratch. They are incrementally completed step by step depending on the modelling methodology. Models evolve during development and are not independent, are interrelated, and in most applications also intertwined. Their interrelationships are often not made explicit and impose changes resulting in inconsistencies to other models due to the variety of models.

Therefore, this paper introduces the theory of model suites as a set of models with explicit associations among the models. Model suites are based on explicit controllers for maintenance of coherence, apply application schemata for their explicit maintenance and evolution, use tracers for establishment of their coherence and thus support co-evolution of information system models. The excitability is captured by integrating model suites and MetaCASE formalisms, exploring the (modelling) method engineering and tool generation required for multi-model development.

Keywords: Model suites, multi-models, model coherence, co-evolution of models, method engineering, MetaCASE.

1 Introduction

1.1 The Evolution from Holistic Modelling to Multi-model Development of Information Systems

Information systems development includes nowadays specification of structuring, functionality, interactivity, components, distribution, etc. One might try to use a holistic

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 314–326, 2010.

approach that incorporates all aspects, facets and concerns into one language. The approach most of developers are choosing is however based on a variety of languages and models. Multi-model specification is currently the state of the art in most sciences. The integration and collaboration of these models is still an open research issue.

Already classical database modelling has been using a three-layer architecture based on a central conceptual schema and associated external schemata and at least one associated internal schema. This architecture may be extended to a multi-tier architecture. Information systems modelling is far more complex since it aims also in representation of functionality, interactivity and distribution.

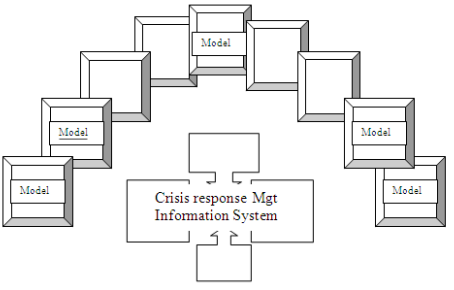
Database modelling is classically defining the database dictionary, database structuring and functionality within one singleton paradigm. This approach has led to sophisticated financial services, to enterprise information systems and other database-backed practical solution which are easy to handle, relatively simple to change and to implement and which satisfied the needs of business in the 90s.

At the same time a number of applications have been developed that used the potential within the data for analysis, for exchange and collaboration of systems, e.g. OLTP-OLAP systems [14], decision support systems, scientific information systems, collaborative information systems and web information systems [18]. These applications do not use a singleton language for data storage, data computation and data delivery. Their languages use different paradigms. We therefore need a way for specification of information systems applications that provide facilities for appropriate modelling depending on the needs.

The language variety of UML models is intriguing and can only be partially handled [17,20]. Many UML diagrams do not support so far a sound foundation and have many different semantics (for instance, UML state charts have 48 different interpretations). Therefore, we need a methodology that supports multi-model development of information systems.

1.2 Exploring Crisis Response Management Information Systems Modelling

An exploratory case study was conducted for a crisis response management system (CRMIS) at an European harbor. This case study asses the extent of complexities' of such systems. Depending on the scale of the disaster, crisis responses in a harbor infrastructure range from small-scale problems, in which a few organizations might be involved, to a full-scale problems, in which multiple organizations are required to resolve and to prevent escalation of the crisis. The (re)design of the CRMIS [3] incorporates the 'virtual team' concept that provides relief-response organizations with a role related picture of the crises development in time critical manner and flexibly satisfies changing information needs. Virtual teams must on-the-fly be extendable when a relief-response organization is required to join relief-response activities. Such teams must remain in employment once its task is completed. Further, it is required to include advanced technologies and available technical infrastructures in a meaningful way for satisfaction of dynamically changing user information needs during a crisis response.



Several types of modelling tools were required to arrive at the systems' architecture. The models interpret the required and relevant solutions of information management, network knowledge and information integration in crisis response management systems designing [9]. A myriad of models were developed: models of knowledge acquisition, knowledge selection, stakeholder analysis, network models for collaboration, coordination models for relief effects, models for knowledge management, models for process descriptions, models for network and node analysis, models for defining knowledge bases and critical knowledge ownerships, and least but not last the typical information system models for data, software, integration, networks, time, space and position, security and technology and a few more.

The diagrams were incrementally developed and models evolved during development. Models were not independent, and were interrelated and in most applications intertwined. Their interrelationships were often not made explicit. Models imposed changes in other models. Changes within one model resulted in inconsistencies to other models due to the variety of models used. Resolutions were time consuming, tedious and led to project delays. Therefore, we introduce an approach to handle co-evolution of information systems modelling as: MetaCASE toolkits for the generation of (information) systems models for a variety of modelling languages, and model suites to assure coherence of co-existence and co-evolution of (information) system models. The application of model suites concept within the disaster management to orchestrate the coherence of model assemblies is available in [22].

1.3 Co-existence and Co-evolution of Models

Multi-model systems development may be based on sequenced development, i.e. at each stage only one model is changed and no other model must be changed correspondingly. This situation is rather idealistic since models reflect different aspects, facets and concern and thus form a co-picture of the entire system. Their association must thus be specified in an explicit form that allows an application to model evolution itself. Additionally we may wish to cope with different abstraction layers such as requirements or conceptual layers, with different abstraction levels such as model, meta-model or meta-meta model level. Furthermore, systems development is a process itself that produces versions, components, sketches and finalised models. The basic rules of multi-model systems development can be summarized in the following fundamental principles: sovereignty of each model, equal existence of each model, and consent about other models. These principles should result in integrated evolution of models and in consentient co-evolution of different models. It follows from the co-existence of different models at a development state that, in principle, they are all equal in status. Therefore, each of the models may evolve on its own right. If however changes in one model have an impact on other models then their associations must be maintained.

Co-existence and co-evolution of models is currently a hot and difficult research topic. The literature is very rich. Typical research problems discussed at present are versioning and evolution of systems [2,8,16], evolution of models themselves [5,23], approaches to refinement of models [7,23], and multi-model management. [17,20].

Our approach is based on a theory, technology and methodology of models suites. This concept allows to integrate different models developed during information systems development and to maintain coherence among models. Since models might be based on different paradigms the association concept must be very flexible. We generalise the concept of institutions that allow to integrate signatures of languages to the concept of model suites which can be based on associations among models either on the signature or the language or the model level. Typically, models are refined in the development process. Associated models must reflect refinements if these have an impact on those models.

1.4 Requirements for Multi-model Information Systems Development

Multi-model information systems development allows to concentrate on one aspects, facet or concern. At the same time, models to be used must be coherent and must co-evolve with the development process. We thus have to meet a number of requirements, e.g. the following main ones:

Problem 1. Explicit specification of model collaboration: Interdependencies among models must be given in an explicit form. The consistency of models must be recursive.

Problem 2. Integrated development of different models: Models are used to specify different views of the same problem or application. They must be used consistently in an integrated form. Their integration must be made explicit. Simultaneous updates of models must be allowed.

Problem 3. Co-evolution of models: Multi-model information systems development must allow data exchange between models and explicit change propagation.

Problem 4. Management of multi-model information systems development: The propagation of changes must be supported by scheduling mechanisms, e.g., ordering of propagation of model changes. The management must support rollback to earlier versions of the model suite. The management should also allow model change during propagation.

This list of requirements may be extended by additional requirements such as (5) combining different representations with mathematical rigor of models, (6) evolution of different representations, (7) version handling for multi-model information systems development, and (8) explicit refinement and abstraction treatment.

1.5 Structure of the Paper

We introduce the concept of model suites in the next section. Section 3 develops an approach to co-evolution of models. Section 4 challenges the concept of model suites in an application environment and demonstrates at the same time how the concept of model suites can be integrated into existing tool environments.

2 Model Suites

2.1 The General Notion of Model Suites

Model suites are an extension of model ensembles [15] used for distributed or collaborating databases [19]. Ensemble databases form a group of databases that support a single effect in an application. Ensemble databases are based on an homogeneous platform and often have a common database modelling language. Model suites also generalise model clusters which are mainly a group of models forming a unit or constituting a collection.

A model suite consists
- of set of models $\{\mathcal{M}_1, ..., \mathcal{M}_n\}$,
- of an association or collaboration schema among the models,
- of controllers that maintain consistency or coherence of the model suite,
- of application schemata for explicit maintenance and evolution of the model suite, and
- of tracers for the establishment of the coherence.

Coherence describes a fixed relationship between the models in a model suite. Two models are coherent when each change in one of the models is propagated to the other model. This change transfer implicitly assumes that the integrity constraints of the corresponding model types remain to be valid. They are non-coherent if there is a random or changing relationship. We aim in an explicit specification of the association schema and use an explicit specification of the *collaboration* among models. For instance, the master-slave association or collaboration propagates any change of the master to its slaves. Slaves do not have any right to change the master without consensus with the master.

2.2 Language Varieties for Models

Typically, a model is defined in a certain language. A model language $\mathcal{L}$ for a model uses some signature $\mathbb{S}$ and a set of constructors $\mathbb{C}$ that allows to build a set of all possible expressions in this language. Typically constructors are defined by structural recursion [21]. The set of constructors may allow to build expressions that do not fulfill certain quality or more generally integrity conditions. Therefore we introduce a set $\Sigma_{\mathbb{S},\mathbb{C}}$ well-formedness conditions.

A model type $\mathcal{T}_{\mathcal{L}_\mathbb{S}} = (\mathcal{L}_\mathbb{S}, \Sigma_{\mathcal{L}_\mathbb{S}})$ is defined by a pair consisting of the language of the model $\mathcal{L}_\mathbb{S}$ of signature $\mathbb{S}$ and by constraints $\Sigma_{\mathcal{L}_\mathbb{S}} \in \mathcal{L}(\Sigma_\mathbb{S}^{\mathrm{WellFormed}})$ applicable to all models defined in the given language.

Model languages $\mathcal{L}_{\mathbb{S}_1}, ...\mathcal{L}_{\mathbb{S}_n}$ may be bound to each other by *partial* mappings $\mathbb{R}_{i,j}$: $\mathcal{L}_{\mathbb{S}_i} \rightarrow \mathcal{L}_{\mathbb{S}_j}$ based on their signatures. These mapping typically define the association of elements among the languages.

A model is based on an expression in the given language. Typically, it has a structure definition, a semantics definition, and a pragmatics definition. Semantics restricts the models we are interested in. Pragmatics restricts the scope of the users of models. We explicitly define a model $\mathcal{M}$ by an expression $struct_\mathcal{M}$ in a language $\mathcal{L}_\mathbb{S}$ that obeys

$\Sigma_{\mathcal{L}_\mathbb{S}}$, by a set of constraints $\Sigma_\mathcal{M}$ defined in the logics of this language. Therefore, each model has its model type. We denote by $\mathcal{M}_\mathcal{T}$ or $\mathcal{M}_i$ for some i the set of all models of this type.

2.3 Model Association and Contracting in Multi-layered Modelling

We want to propagate changes in one model to other models. These associated changes can be explicitly modelled by a *collaboration contract* among models. We distinguish three facets of collaboration: communication, coordination and cooperation. *Communication* is used in a variety of facets as an act or instance of transmitting or a process by which information is exchanged between models through a common system. *Coordination* expresses the act or action of coordinating the harmonious functioning of models for effective results. *Cooperation* expresses the action of cooperating.

The collaboration style of a model suite is based on four components describing

supporting programs of the information system including session management, user management, and payment or billing systems;

data access pattern for data *release* and *locking* through the net, e.g., broadcast or P2P, for *sharing* of resources either based on transaction, consensus, and recovery models or based on replication with fault management, and for *remote access* including scheduling of access;

the style of collaboration on the basis of peer-to-peer models or component models or push-event models which restrict possible communication;

and the coordination workflows describing the interplay among parties, discourse types, name space mappings, and rules for collaboration.

Collaboration pattern generalize protocols and their specification [13]. We know a number of collaboration pattern supporting *access and configuration* (wrapper facade, component configuration, interceptor, extension interface), *event processing* (reactor, proactor, asynchronous completion token, accept connector), *synchronization* (scoped locking, strategized locking, thread-safe interface, double-checked locking optimization) and *parallel execution* (active object, monitor object, half-sync/half-async, leader/followers, thread-specific storage):

Proxy collaboration uses partial system copies (remote proxy, protection proxy, cache proxy, synchronization proxy, etc.).

Broker collaboration supports coordination of communication either directly, through message passing, based on trading paradigms, by adapter-broker systems, or callback-broker systems.

Master/slave collaboration uses tight replication in various application scenarios (fault tolerance, parallel execution, precision improvement; as processes, threads; with(out) coordination).

Client/dispatcher collaboration is based on name spaces and mappings.

Publisher/subscriber collaboration is also known as the observer-dependents paradigm. It may use active subscribers or passive ones. Subscribes have their subscription profile.

Model/view/controller collaboration is similar to the three-layer architecture of database systems. Views and controllers define the interfaces.

A contract $\mathfrak{C}$ consists of a declaration of constraints, of a description of the enforcement mechanism and of a prescription of modification steps that transform a coherent model suite into a coherent model suite.

A contract may include obligations, permissions and sanctions. Therefore,

- contracts declare correctness of a model suite, separate exceptional states from normal states for these model suites, and forbid meaningless model suites,
- contracts enable the direct manipulation of the model suite as transparently as possible and offer the required feedback in the case of invalidation of constraints based on echo back, visualisation of implications, on deferred validation, instant projection and hypothetical compilation, and
- contracts consider mechanisms that address the long term coherence of a model suite by forecasting confirmation, by anticipating changes made in a team, by providing a mechanism for adjusting and confirming correctness, and by specifying diagnostic queries for inspection of model suites.

2.4 The Notion of the Model Suite

A model suite type $\mathcal{ST} = (\mathcal{T}_{\mathcal{L}_{\mathbb{S}_1}}, ..., \mathcal{T}_{\mathcal{L}_{\mathbb{S}}}, \Sigma_{\mathcal{L}_{\mathbb{S}_1},...,\mathcal{L}_{\mathbb{S}_n}})$ is given by model types $\mathcal{T}_{\mathcal{L}_{\mathbb{S}_i}}$ defined on a set $\mathcal{L}_{\mathbb{S}_i}, ..., \mathcal{L}_{\mathbb{S}_n}$ of languages and a set $\Sigma_{\mathbb{S}_1,...,\mathbb{S}_n}$ of constraints on these languages.

A model suite $\mathfrak{S}$ on a model suite type $\mathcal{ST}$ consists of models $(\mathcal{M}_1, ..., \mathcal{M}_n)$ of type $\mathcal{T}_{\mathcal{L}_{\mathbb{S}_i}}$ that obey $\Sigma_{\mathcal{L}_{\mathbb{S}_1},...,\mathcal{L}_{\mathbb{S}_n}}$.

The contract on $\mathfrak{C}$ thus consists of the constraints $\Sigma_{\mathcal{L}_{\mathbb{S}_1}} \cup ... \cup \Sigma_{\mathcal{L}_{\mathbb{S}_n}} \cup \Sigma_{\mathcal{L}_{\mathbb{S}_1},...,\mathcal{L}_{\mathbb{S}_n}}$, a description of the enforcement mechanisms for any operation that can be used for modification of one model, and a set of consistent evolution transformations.

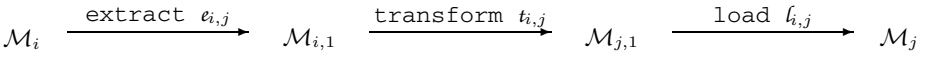
We use approaches developed for control theory for handling dynamics of model suites. Dynamics of layered systems is defined by pending objects, i.e. request results in initialisation of a new pending object, request being processed, and pending object issues a new request. The new request may die after issuing the request or may wait for the response for a certain time slot with cancellation activities or may wait for the response that the request has been accepted or that is request is processed or that the request got an answer.

3 Co-evolution of Information Systems Models Based on Model Suites

Synchronisation of models is a difficult matter for which a general solution is unlikely to exist. Instead of the category-based framework for direct synchronisation [6] we generalise the database approach. Heterogeneous models $\mathcal{M}_1, \mathcal{M}_2$ that should be synchronised consist of converging submodels, i.e., $\mathcal{M}_1 = \mathcal{M}_{1,0} \Cup \mathcal{M}_{1,2}$ and $\mathcal{M}_2 = \mathcal{M}_{2,0} \Cup \mathcal{M}_{1,2}$ with $\mathcal{M}_{1,0} \curlyvee \mathcal{M}_{1,2}$ and $\mathcal{M}_{2,0} \curlyvee \mathcal{M}_{1,2}$[1]. Moreover this model is limited by the assumption that the submodel $\mathcal{M}_{1,2}$ is common for both models. For heterogeneous

[1] $\Cup$ denotes the generalised union of models, $\curlyvee$ denotes separatability or divergency of models, and $\boxtimes$ denotes the generalised join.

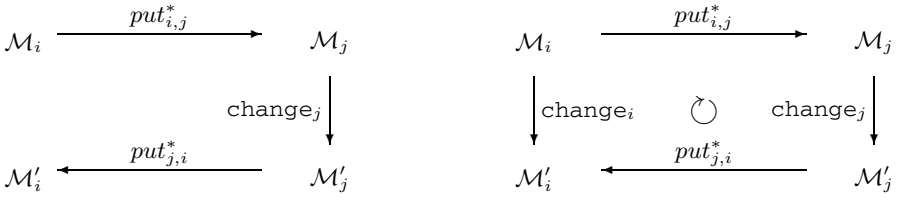
models we assume that $\mathcal{M}_i = \mathcal{M}_{i,0} \boxtimes \mathcal{M}_{i,1}$ from $\mathfrak{L}_i$ and $\mathcal{M}_j = \mathcal{M}_{j,0} \boxtimes \mathcal{M}_{j,1}$ from $\mathfrak{L}_j$ for models $\mathcal{M}_i, \mathcal{M}_j$ that are going to be synchronised. Given furthermore a mappings $t_{i,j} : \mathcal{M}_{i,1} \longmapsto \mathcal{M}_{j,1}$ $t_{j,i} : \mathcal{M}_{j,1} \longmapsto \mathcal{M}_{i,1}$ for which extensions of $\mathbb{R}_{i,j}, \mathbb{R}_{j,i}$ exist in $\mathfrak{L}_i$ and $\mathfrak{L}_j$, respectively.



The product $e_{i,j} \circ t_{i,j} \circ l_{i,j}$ of the mappings is denoted by $put_{i,j}$. This product is neither left-inverse to $put_{j,i}$ nor must have a right-invers $put_{j,i}$. This phenomenon is well known for updates of views in databases [1,10]. Since models must obey integrity constraints of their types we might have models for which $put_{i,j}$ is not defined. Two models $\mathcal{M}_i$ and $\mathcal{M}_j$ are called *coexisting* if $put_{i,j}(\mathcal{M}_i) \preccurlyeq \mathcal{M}_j$ and $put_{j,i}(\mathcal{M}_j) \preccurlyeq \mathcal{M}_i$. We observe that a model $\mathcal{M}_i$ may have many coexisting models $\mathcal{M}_j$.

The mappings $put_{i,j}$, $e_{i,j}$, $t_{i,j}$, and $l_{i,j}$ may be generally given for the set of all models defined on the model types $\mathcal{T}_{\mathcal{L}_{\mathbb{S}_i}}$ and $\mathcal{T}_{\mathcal{L}_{\mathbb{S}_j}}$. In this case the sub-model embedding must be canonical.

We observe that a model $\mathcal{M}_i$ may have many coexisting models $\mathcal{M}_j$. If we use a canonical embedding then the mappings $put_{i,j}$ can be defined on the basis of the constant complement [1,10], i.e., $\hbar(\mathcal{M}_j, i) = \mathcal{M}_j \boxminus e_{j,i}(\mathcal{M}_j)$. We may now extend the mapping $put_{i,j}$ by the constant complement of the range model and define an integration condition by $\mathcal{M}_j = put^*_{i,j}(\mathcal{M}_i, \hbar(\mathcal{M}_j, i))$. If the integration condition is valid for coexisting models then we may support also changes in one model and propagate the changes to the other model.



We require that the mappings put^* are well-behaved, i.e.

$put^*_{j,i}(put^*_{i,j}(\mathcal{M}_i, \hbar(\mathcal{M}_j, i)), \hbar(\mathcal{M}_i, j))$ is defined and
$put^*_{j,i}(put^*_{i,j}(\mathcal{M}_i, \hbar(\mathcal{M}_j, i)), \hbar(\mathcal{M}_i, j)) = \mathcal{M}_i$.

The coexistence of models is not sufficient for change propagation. If $\mathcal{M}_j$ is changed to $\mathcal{M}'_j$ by the change operation $\texttt{change}_j$ then the change diagram should commute, i.e. this change operation has a related change operation $\texttt{change}_i$ that allows to change $\mathcal{M}_i$ directly to $\mathcal{M}'_i$. The change operation is typically defined on two arguments: the original model $\mathcal{M}_j$ and an auxiliary model $\mathcal{M}_{aux}$. Model suite change is called *synchronised* for i, j and a set $\mathcal{O}_j^{change}$ of change operations defined on $\mathcal{M}_j$ if for each change operation $o_j(\mathcal{M}_j, \mathcal{M}_{aux})$ from $\mathcal{O}_j^{change}$ a change operation $o_i(\mathcal{M}_i, \mathcal{M}_{aux})$ in the set $\mathcal{O}_i$ of operations on $\mathcal{M}_i$ exists so that the change diagram commutes for the same auxiliary model $\mathcal{M}_{aux}$, i.e., $o_i(\mathcal{M}_i, \mathcal{M}_{aux}) = put^*_{j,i}(\mathcal{M}'_j, \hbar(\mathcal{M}_i, j))$ for $\mathcal{M}'_j = o_j(put^*_{i,j}(\mathcal{M}_i, \hbar(\mathcal{M}_j, i)), \mathcal{M}_{aux})$.

The complement should be constant. Therefore we may use $\hbar(\mathcal{M}_i, j)$ instead of $\hbar(o_i(\mathcal{M}_i, \mathcal{M}_{aux}), j)$.

Change operations are used for model change and evolution. The order of changes is typically important. We call two change operations o_i, o_j *liberal* to the models $\mathcal{M}_i, \mathcal{M}_j$ if $\mathcal{M}'_i = put^*_{j,i}(\mathcal{M}'_j, \hbar(\mathcal{M}_i, j))$ and $\mathcal{M}'_j = put^*_{i,j}(\mathcal{M}'_i, \hbar(\mathcal{M}_j, i))$ for $\mathcal{M}'_i = o_i(\mathcal{M}_i, \mathcal{M}_{aux})$ and $\mathcal{M}'_j = o_j(\mathcal{M}_j, \mathcal{M}_{aux})$. Liberality can be extended to confluence and Church-Rosser properties [21]. Liberal change operations allow to change on model and then to apply all the changes to coherent model suites.

Contract management becomes in this case rather simple. Enforcement may directly be applied to all coexisting models. We also may restrict change operation by `no action`, `cascade`, `oblige` enforcements. 'No action' means that if a change operation cannot propagated to the other model then this change operation is rolled back. Cascade enforcement requires that the other model must be changed as well. Oblige enforcement allows to delay the change operation on the other model to a later stage.

Typically the set of change operations is defined by a change pattern. For instance, all database changes can be defined through *insert*, *delete* and *update*. [21] defines a small set of change pattern for extended ER language schemata that allow to express any change in HERM schemata.

Finally we may also require that the `undo` operation is also supported. This operation is nothing else than another `change` operation that results in restoring the old model.

4 Tool Support

4.1 The MetaCASE Toolkit

We introduce an approach to handle co-evolution of information systems modeling as: MetaCASE toolkits for the generation of (information) systems models for a variety of modeling languages, and model suites to assure coherence of co-existence and co-evolution of (information) system models.

MetaCASE has the capacity to generate toolkits for any modelling language. The fundamental theory behind MetaCASE is the separation of modelling concepts from their visual representations [4]. The set of concepts that can produce all expressions of a language is combined with graphic representations to function as a modelling tool. The model visualizes a graphical representation of a real world situation. According to [4] the central repository of the MetaCASE is a layered database architecture consisting of four layers: signature, language, model and data. The collection of modelling constructs ($\mathcal{MC}$) that allows building a set of all expressions in an arbitrary modelling language belongs to the signature layer.

The modelling constructs are elements of modelling languages. The modelling construct signature ($\mathbb{MCS}$) forms the hart of the signature layer. $\mathbb{MCS}$ is a collection of modelling constructs ($\mathcal{MC}$), constraints or rules of modelling constructs $\Sigma_{\mathcal{MC}}$ and a set of derivation rules for models' population $R(\mathcal{M})$. This approach supports a high level of conceptuality and a sufficient comprehensibility in order to produce executable models and in order to be extendable.

Modelling constructs ($\mathcal{MC}$) consist of basic elements for modelling arbitrary modelling languages:

(I) The set $\mathcal{P}$ of *roles* describes predicators or associations.
(II) The non-empty finite set $\mathcal{OT}$ of *object types* contains disjoint subsets of
- label types ($\mathcal{LT}$), entity types ($\mathcal{ET}$),
- collection types ($\mathcal{GT}$), sequence types ($\mathcal{ST}$) of sets or sequences of object types, correspondingly,
- relationship types that form a partition F of the set P, and
- MetaCASE model types $\mathcal{MT}$ that are decomposed into modelling constructs.

(III) *Functions:* We are given at least the following functions:
- The function $Base$: $\mathcal{P} \rightarrow \mathcal{OT}$ associates roles to object types.
- The function Elt: $\mathcal{GT} \cup \mathcal{ST} \rightarrow \mathcal{OT} \setminus \mathcal{LT}$ yields the elements of collection types and sequence types.

(IV) The *relation schema* defines object types for models, i.e. a subset of $\mathcal{MT} \times \mathcal{OT}$. This relation describes the elements in models.
(V) *Specialization* $Spec$ and *generalization relations* Gen as subsets of $\mathcal{ET} \times (\mathcal{OT} \setminus \mathcal{LT})$.
(VI) The *many sorted algebra* $D =< \mathcal{D}, \mathcal{F} >$ with a set of concrete domains $\mathcal{D}$ used to instantiate label types (e.g. strings, natural numbers) and a set of operations $\mathcal{F}$ (e.g. $+$).
The function $Dom : \mathcal{LT} \rightarrow \mathcal{D}$ yields the domain of label types.

Formal details are avoided for simplicity and we refer to [4] for MetaCASE and [11] for detail formalism of Predicate Set Model (PSM).

To assure the functioning and operation of MetaCASE the signature layer further consist of signature constructs for storage and manipulations, versions , views, integrity and consistency, concurrency controls, security, distribution, control of integrity and interfaces. The signature constructs are orthogonal and modular components contributing to agility, extensibility and reusability. The picture below depicts the constituents of the signature, language layer, model, and the data layer as well as the mappings between these layers and finally the incremental layering of these layers.

This layering is typical for data-centric model suites. Models are first developed for the structure of the data and the integrity constraints at this layer. Functionality is typically built in information systems development on top of the structuring specification. Workflow or method editors can thus be developed on top of structuring specification. The signature layer is used to compile these layers into a common language. Therefore, the signature layer combines models within a model suite into a holistic specification. It also allows to co-evolve such models.

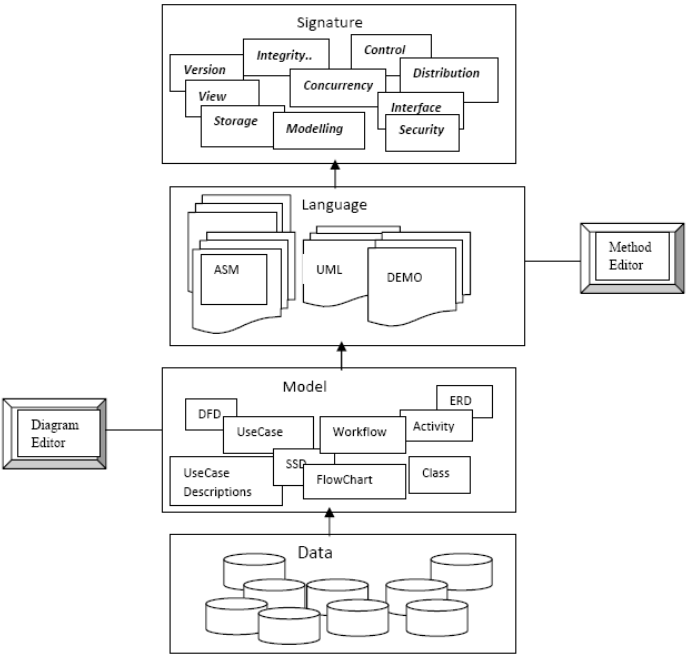
Therefore, our MetaCASE realisation supports a model suite evolution with a central signature, a set of models, a set of model association schemata, a set of contracts among these models that support control of coherence within such model suites, an explicit application schema that is ruled by MetaCASe, and also inherent tracers for model coherence.

The language layer represents models of modelling languages consisting of constructs, rules and behavior normally called the (meta) models. The language layer is for the construction of (meta) models of languages and for the generation of modelling tools. This activity is called the (modelling) method engineering. The model layer accommodates language expressions; the models which are views of the solution, and are

the information systems models. The populations of model instances are in the data layer.

4.2 Model Suite Support in MetaCASE

The main advantages of MetaCASE solutions are: the ability to configure toolkits for structure oriented modelling languages (e.g. ERD, Domain Class diagrams etc.), behavior oriented modelling languages (e.g. event charts, use cases, Petri nets, etc.), and for process oriented modelling language (e.g. Activity diagrams, Systems Sequence diagrams, DEMO etc.), the reusability, agility and extendibility to complex modelling requirements such as model suites [4].



The design, development and maintenance of complex systems such as CRMIS challenge current systems modelling by requiring a variety of tools that are capable of managing a number of changes in scope, impact, granularity, abstraction level etc. embedded in modelling support. This challenge can be met by integrating model suites into the modelling constructs of the MetaCASE. The main construction is given by:

– A model suite type $\mathcal{ST}$ is always decomposed into MetaCASE model types $\mathcal{MT}$.

Model suites $\mathcal{M}_1, ..., \mathcal{M}_n$ are sets of models with explicit associations among the models, with explicit controllers for maintenance of coherence of the models, with application schemata for their explicit maintenance and evolution, and traces for establishment of their coherence as defined in Sections 2. Thereby we achieve a formal foundation for arriving at a holistic approach for multi-model development, management, and maintenance of information systems models that provides a holistic and powerful solution to the four problem areas [12] where research and tools should be extended: the representational, conceptual, methodological, and implementation areas.

5 Conclusion

This paper introduces the conception of model suites for support of multi-model information systems development. Model suites allow to maintain coherence of models during evolution of some of its models. Coherence is based on associations among models, controllers for maintenance of consistency within a model suite, an application schema for handling maintenance and evolution of schemata, and tracers for establishment of coherence.

The conception of model suites explicitly assumes a constructive and compositional approach to modelling. In this case, elements can be separated into signatures, languages, models and populations (or databases). The approach is sufficient for most modelling languages such as UML diagrams, extended ER models, object-relational models, network models, or XML models.

Since the model suites concept is integrated into the MetaCASE's signature layer, the toolkits generated within a MetaCASE in combination with the model suite concept becomes a powerful multi-model development environment. Such MetaCASE model suite toolkits can exhibit following characteristics:

1. Explicit specification of model suite collaboration: Interdependencies among models can be given in an explicit form. The consistency of models becomes recursive.
2. Integrated development of different models: Models are used to specify different views of the same problem or application. They become consistent in an integrated manner. Their integration is made explicit. Simultaneous updates of models are allowed.
3. Co-evolution of models: The model suites allow data exchange between models. The changes within one model are propagated to all dependent models.
4. Combining different representations with mathematical rigor of models: Each model consists of well-defined semantics as well as a number of representations for the display of model content. The representation and the model are tightly coupled.
5. Evolution of different representations: Changes within any model either be refinements of previous models or explicit revisions of such models. These changes are enforced for other representations as well whenever those concerns occur.
6. Management of model suites: The propagation of changes are supported by scheduling mechanisms, e.g., ordering of propagation of model changes. The management must support rollback to earlier versions of the model suite. The management should also allow model change during propagation.
7. Version handling for model suites: Model suites may have different versions.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of databases. Addison-Wesley, Reading (1995)
2. Bounif, H.: Data model versioning and database evolution. In: Encyclopedia of Database Technologies and Applications, pp. 110–115. Idea Group (2005)
3. Chen, N., Dahanayake, A.N.W.: Role-based situation-aware information seeking and retrieval for crisis response. Journal of Intelligent Control Systems 12(2), 186–197 (2007)
4. Dahanayake, A.N.W.: An Environment to support flexible information modeling. PhD thesis, Delft University of Technology, The Netherlands (1997)

5. Danoch, R., Shoval, P., Balaban, M.: Hierarchical evolution of entity-relationship diagrams - a bottom-up approach. In: Proc. EMMSAD 2001 Workshop associated with CAiSE 2001, Interlaken (2001)
6. Diskin, Z.: Algebraic models for bidirectional model synchronization. In: Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., Völter, M. (eds.) MODELS 2008. LNCS, vol. 5301, pp. 21–36. Springer, Heidelberg (2008)
7. Egyed, A.: Consistent adaptation and evolution of class diagrams during refinement. In: Wermelinger, M., Margaria-Steffen, T. (eds.) FASE 2004. LNCS, vol. 2984, pp. 37–53. Springer, Heidelberg (2004)
8. Franconi, E., Grandi, F., Mandreoli, F.: Schema evolution and versioning: A logical and computational characterisation. In: Balsters, H., De Brock, B., Conrad, S. (eds.) FoMLaDO 2000. LNCS, vol. 2065, pp. 85–99. Springer, Heidelberg (2001)
9. Harnesk, D., Linsdtrom, J., Samuelsson, S.: Socio-technical design approach for crisis management information systems. International Journal of Information Systems for Crisis Response and Management 1(3), 1–8 (2009)
10. Hegner, S.J.: Information-optimal reflections of view updates on relational database schemata. In: Hartmann, S., Kern-Isberner, G. (eds.) FoIKS 2008. LNCS, vol. 4932, pp. 112–131. Springer, Heidelberg (2008)
11. Hofstede, A.H.M.: Information Modeling in Data Intensive Domains. PhD thesis, Katholike University of Nijmegen, The Netherlands (1993)
12. Kelly, S., Smolander, K.: Evolution and issues in MetaCASE. Information & Software Technology 38(4), 261–266 (1996)
13. König, H.: Protocol Engineering: Prinzip, Beschreibung und Entwicklung von Kommunikationsprotokollen. Teubner, Stuttgart (2003)
14. Lewerenz, J., Schewe, K.-D., Thalheim, B.: Modeling data warehouses and OLAP applications by means of dialogue objects. In: Akoka, J., Bouzeghoub, M., Comyn-Wattiau, I., Métais, E. (eds.) ER 1999. LNCS, vol. 1728, pp. 354–368. Springer, Heidelberg (1999)
15. Prochnow, S., von Hanxleden, R.: Statechart development beyond WYSIWYG. In: Engels, G., Opdyke, B., Schmidt, D.C., Weil, F. (eds.) MODELS 2007. LNCS, vol. 4735, pp. 635–649. Springer, Heidelberg (2007)
16. Proper, H.A.: A Theory for Conceptual Modelling of Evolving Application Domains. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands (1994)
17. Salay, R., Mylopoulos, J., Easterbrook, S.M.: Using macromodels to manage collections of related models. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 141–155. Springer, Heidelberg (2009)
18. Schewe, K.-D., Thalheim, B.: Reasoning about web information systems using story algebras. In: Benczúr, A.A., Demetrovics, J., Gottlob, G. (eds.) ADBIS 2004. LNCS, vol. 3255, pp. 54–66. Springer, Heidelberg (2004)
19. Schewe, K.-D., Thalheim, B.: Development of collaboration frameworks for web information systems. In: IJCAI 2007 (20th Int. Joint Conf on Artificial Intelligence, Section EMC 2007 (Evolutionary models of collaboration), Hyderabad, pp. 27–32 (2007)
20. Schmidt, P., Thalheim, B.: Management of UML clusters. In: Abrial, J.-R., Glässer, U. (eds.) Rigourous Methods for Software Construction and Analysis. LNCS, vol. 5115, pp. 111–129. Springer, Heidelberg (2009)
21. Thalheim, B.: Entity-relationship modeling – Foundations of database technology. Springer, Berlin (2000)
22. Thalheim, B.: Model suites. In: Jaakkola, H. (ed.) Selected Topics on Distributed Disaster Management: Towards Collaborative Knowledge Clusters, pp. 108–128. Tampere University Press, Porin yksikkö (2008)
23. Wachsmuth, G.: Metamodel adaptation and model co-adaptation. In: Ernst, E. (ed.) ECOOP 2007. LNCS, vol. 4609, pp. 600–624. Springer, Heidelberg (2007)

Process Line Configuration: An Indicator-Based Guidance of the Intentional Model MAP

Rébecca Deneckère and Elena Kornyshova

Centre de Recherche en Informatique
Université Paris 1 Panthéon-Sorbonne, 90 rue de Tolbiac, 75013 Paris, France
{rebecca.deneckere,elena.kornyshova}@univ-paris1.fr

Abstract. Variability has proved to be a central concept in different engineering domains to develop solutions that can be easily adapted to different organizational settings and different sets of customers at a low price. The MAP formalism has a high level of variability as it is expressed in an intentional manner through goals and strategies. However, a high level of variability means a high number of variation points. A process customization is then required to offer a better guidance. The Product lines have appeared with this management of variability and customization. Furthermore, we propose the Process line concept to represent the processes that may be customized to a given project. Our goal is to enhance the Map guidance by specifying the *MIG* (*M*ap *I*ndicator-based *G*uidance) approach. We suggest several guidance approaches based on an indicators' typology. We illustrate our proposal with an example from the requirement engineering field.

Keywords: Indicator, MAP, Process guidance, Process line, Configuration.

1 Introduction

Over the decades, variability in Software Engineering has become increasingly important. At the beginning, a system met the purpose of a single organization and of a simple set of customers, whereas nowadays, a system must be conceived in a larger perspective, to meet the purpose of several organizations and to be adaptable to different usage situations and customer sets [1]. The variability is the ability to be subject to variation. As a result, the notion of software variability is defined as the ability of a software system to be changed, customized or configured to a specific context [2]. Whereas the software community studies variability as a design problem and concentrates on implementation issues [4] [5] [6], we believe like [7] that capturing variability at the goal level is essential to meet the multi-purpose nature of these new information systems. They incorporate variability in the functionality that they provide and are able to self adapt to the situation at hand.

The increasing variability in software engineering has led to the establishment of the concept of *Product lines*. Product line engineering is a paradigm to develop software applications using platforms and mass customization, which means that the commonalities and the differences in the applications of the product line have to be modeled in a common way [3]. As a Product may be envisioned as a specific customization of a

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 327–339, 2010.

Product line, a Process may also be seen as a specific customization of a *Process line*. This ability to derive a process configuration from common characteristics in a repeatable manner is based on the variability of the process models. This runtime configuration increases the context-awareness of the processes. Configurable process models has already been put forward in [8]. This kind of models is defined by combining similar business process models within a family, thus creating process variability. The process family is then configured to fit the requirements of specific organizations or projects. However, [8] aims at creating configurable processes in the particular field of business process management.

In our proposition, we use methodological process models which already have a high level of variability as they are goal-oriented. Goal modeling has been found to be an effective way for identifying requirements of software systems by focusing on understanding the intentions of the involved stakeholders [9] [10]. The process model MAP [1] [11] is an example of goal model conceived to meet this challenge. A Map expression provides a synthetic view of the process variability in an easy way to understand. Variations are revealed in two ways, by the gradual movement down the different levels of a top map, and by the alternative paths available at a given map level [1]. [12] detailed this intrinsic variability of the MAP model.

A high level of variability creates an increased need for guidance. As the engineer go through its process, he reaches variation points where more information is needed to make further decisions. An enhanced version of the MAP model has been described in [13] with the integration of weight values in order to use the graph theory for Map guidance. In our proposal, we suggest a more complete Map Indicator-based Guidance (MIG) approach, which aims at enhancing the MAP guidance by representing it as a configurable process within the Process line concept. The MIG approach has three main properties: (i) it is viewed as a decision-making problem; (ii) it includes the context indicators typology, and (iii) it suggests different kind of guidance based on indicators. Firstly, as the Map model has an intentional nature, it requires decisions in its navigation. Secondly, an indicator typology is suggested for making guidance decisions. This typology is adapted to the MAP model, with indicators deduced from either the Map arguments or the project situation. Thirdly, MIG contains three approaches allowing different guidances based on the proposed typology. The MIG approach increases the context-awareness with the use of context indicators which express runtime information about the project at hand. These indicators guide the engineer during the process execution and help him to select the more adequate path in a Map, i.e. to configurate his process. This dynamic configuration allows having slightly different process instances, based on external settings, from the same process model.

The MAP process line is described in Section 2. Section 3 defines the process line configuration and Section 4 specifies the approaches to use it. We conclude in Section 5.

2 The MAP Process Line

The MAP model has been introduced in the Information System Engineering domain [11] and validated in several fields, as requirement engineering [14], method engineering [15] or process modelling [11]. Maps are representations of processes. As process models, they can be compared to the various types of process modeling languages and

formalisms. These formalisms can be classified according to their orientation to activity-sequence oriented, agent-oriented, state-based, and intention-oriented languages [16]. However, most process modeling languages do not employ a goal construct as an integral part of the model. These models focus on how the process is performed and externalize what the process is intended to accomplish in the goal. In contrast, intention-oriented process modeling focuses on what the process is intended to achieve, i.e. why the process is performed. This intentional level is used to guide engineers through IS processes by dynamic choices of the tasks sequences. Each time that an intention is reached, the model suggests the tasks that can be executed on the next step. As a result, the concrete process is not rigid but constructed dynamically following the situation. In this view, the MAP process model may be considered as a process line, customizable on the fly. An example from the *Crews-L'écritoire* approach is given in Fig. 1. This Map has been described in [17].

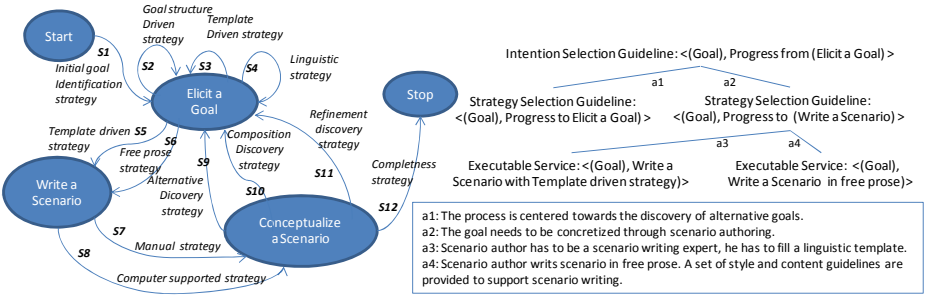


Fig. 1. A Map example: The Crews-L'écritoire approach (CL Map) and examples of its Intention Selection Guidelines and Strategy Selection Guidelines

A map is presented as a diagram where nodes are *intentions* and edges are *strategies*. The directed nature of this diagram shows the precedence links between intentions. An edge enters a node if its associated strategy can be used to achieve the target intention (the given node). Since there can be multiple edges entering a node, a map is able to represent the many ways for achieving an intention. The figure 2.A (see section 3) shows the MAP model using the UML formalism. The key notion of a map is a *section* which is an aggregation of a *source intention* and a *target intention*, linked together with a *strategy*. It embeds the knowledge corresponding to a particular process to achieve the target intention from a specific situation following a particular technique.

Sections in a map are related to each other by three kinds of relationships namely thread, path and bundle. A thread relationship shows the possibility for a target intention to be achieved (from the same source intention) in several ways. Each of these ways is expressed as a section in the map. A path relationship establishes a precedence relationship between sections. For a section to succeed another, its source intention must be the target intention of the preceding one. No path is predefined as the engineer constructs his own path following the situation at hand. Given the thread and the path relationships, an intention can be achieved by several combinations of sections. A bundle relationship shows the possibility for several sections having the same source and target intentions to be mutually exclusive.

A refinement relationship shows that a section of a map can be refined as another map. Refinement is an abstraction mechanism by which a complex assembly of sections at level *i+1* is viewed as a unique section at level *i*. This relationship introduces levels in the process representation as each map may be represented as a hierarchy of maps.

An *Executable Service* is linked to each section. It provides an operational means to fulfill the intention. It implies the transformation of the product under development with the execution of a service (which can be a guideline, a workflow, an algorithm, etc.).

There are two guidelines associated to each section. The *Intention Selection Guideline* (ISG) identifies the set of intentions that can be achieved in the next step and proposes arguments to select one of them. The *Strategy Selection Guideline* (SSG) completes the first one as it determines all the strategies that can be used to achieve the target intention. Fig. 1 shows an example of these two kinds of guidelines applied on the CL map. In this example, we have an ISG and a SSG. The ISG allows selecting between two intentions *"Elicit a Goal"* and *"Write a Scenario"* from the intention *"Elicit a Goal"*. The arguments *a1* and *a2* are used to guide the selection. If the *"Elicit a Goal"* intention is selected (a1), the *Executable service* associated with the section is used by the enactment mechanism to achieve the target intention. If the *"Write a Scenario"* intention is selected (a2), the SSG is then used. It allows selecting a strategy between *"template driven"* (a3) and *"free prose"* (a4).

These arguments guide the engineer to navigate in the Map. Our goal is to offer a way to formalize these arguments, written in natural language, and other information, into indicators, in order to be able to customize the process to the project at hand.

3 MAP Process Line Configuration: The MIG Indicators Typology

The MIG indicators typology is based on the characteristics of IS development projects [15]. A study of these known characteristics has allowed to deduce a set of indicators and their possible values. This formalization improves the characteristics usability as automatic guidance is then possible. The Fig. 2.B shows the MAP model enhanced with indicators. Each indicator is connected to different MAP concepts. The arguments (ISG and SSG) have been also deleted. Indeed, the use of the typology allows the engineer to transform arguments, initially defined in natural language, into indicators values. Some indicators may be used whichever the process model used (generic indicators) whereas other are specific for the MAP model (specific indicators). Indicators can be quantitative or qualitative. The use of values increases the possibility of automatic guidance.

Generic indicators covers essential aspects of IS engineering projects. Based on [15], they are classified into four facets: organizational, human, application domain, and development strategy (cf. Appendix B). The *organizational* facet highlights organizational aspects of the IS project (like *Size*, *Cost*…). The *human* facet deals with the qualities of the persons involved (as *Expertise degree*, *User involvement*…). The *application domain* facet includes indicators characterizing the domain of IS project (like its *Complexity*, *Variability*…). Indicators gathering different characteristics of development strategy are organized into the *development strategy* facet (as the *Number of goals*, the *Project organization*…). These indicators may be applied at different level of granularity on the MAP model. For instance, through the navigation,

indicators may be used to choose the next *intention* to realize or the specific *strategy* to reach this intention. As a section represent a specific way to reach an intention following a particular strategy, the indicators applied on either of these two concepts may have an impact on the *section*. On the same way, as a Map is composed of sections, some of their indicators may be applied to the *Map* level.

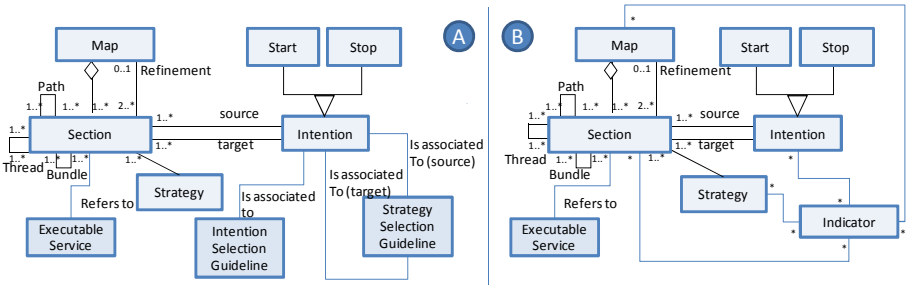


Fig. 2. MAP model (A) and MAP process line (B)

Specific Map indicators are evaluated 'on the fly' (Table 1). They concern two MAP concepts: Intention and Section, and take into account the evolving situation of the product under construction. For instance, if the execution of a section does not completely realize the target intention, it will then be necessary to execute a reflexive section on this intention, in order to complete it (Goal achievement degree). The indicators provide also an evaluation of the section completeness. Following the situation of the product, the section may be executed several times in order to realize completely the guideline (Section completeness degree). Two other indicators, instead of testing the achievement or completeness of these concepts, take into account the satisfaction degree. For instance, even if an intention is not fully realized, the engineer may decide that the realization is satisfactory and go further in the navigation.

The specific indicators are not applied through the composition link to the Map itself as they are dynamic and reevaluated after each execution of a section.

Table 1. Specific Map Indicators

Indicator	Value domain		Connections			
	Quantitative	Qualitative	Map	Int	Str	Sect
Goal satisfaction degree	3-grade scale.	ENUM:{low, normal, high}		X		X
Goal achievement degree	3-grade scale	ENUM:{low, normal, high}		X		X
Section satisfaction degree	3-grade scale.	ENUM:{low, normal, high}				X
Section completeness degree	3-grade scale	ENUM:{low, normal, high}				X

Arguments on the Map *Vs* Indicators. Two kinds of guidelines (ISG and SSG) provide arguments to help the engineer to make his choice. An analysis of these arguments allows the engineer to fill some indicator values. For instance, the arguments of the CL Map on Fig.1 may be translated as follows on the Map concepts (Table 2). See that the *Expert role* Indicator has the same value. The reason is that the CL Map has been defined for analysts of the requirement engineering domain. As a result, all Map intentions have the 'Analyst' value for this indicator.

Table 2. Translation of CL Map Arguments

Arguments	Indicators	Intention	Strategy	Section
a1: The process is centered towards the discovery of alternative goals.	Expert role	Analyst		Analyst
	Goal achievement degree	Low		Low
a2: The goal needs to be concretized through scenario authoring.	Expert role	Analyst		Analyst
	Goal achievement degree	High		High
a3: Scenario author has to be a scenario writing expert, he/she has to fill a linguistic template.	Expert role	Analyst		Analyst
	Expertise degree		3	3
a4: Scenario author writes scenario in free prose. A set of style and content guidelines are provided to support scenario writing.	Expert role	Analyst		Analyst
	Expertise degree		1	1

The *Expertise degree* indicator is used in *a3* and *a4*. The corresponding Map concept is the Strategy (and, by relationship, the Section). *Fill a template* means a high value of expertise degree (value=3), whereas *writing a text in free prose* requires a low value of the same indicator (value=1). The *Goal achievement degree* is used in *a1* and *a2*. Its value is Low if the goal has not been completely achieved and that the engineer has to execute a section with the same target intention (to make a cycle), whereas its value is High if the goal has been entirely achieved (so the navigation may reach another intention).

Application of the typology to the CL Map. To illustrate our proposal, we applied the previously defined typology to the CL Map of Fig. 1. In this example, we have selected six indicators: *Expert role*, *Expertise degree*, *Formality degree*, *Complexity degree*, *Relationships degree* and *Goal achievement degree*. Table 3 illustrates these indicators applied to the concepts Intention, Strategy and Section.

Intentions indicators. The two intentions Start and Stop don't have indicators as they only represent the beginning and ending of the process. As explained before, the three other intentions have all the same value for the *Expert Role* ('Analyst').

Strategies indicators. On the six selected indicators of the typology, only four may be applied to strategies: *Expertise degree*, *Formality degree*, *Relationship degree* and *Complexity degree*. The first two indicators have quantitative values between 1 and 3. For instance, if the value of *Expertise degree* is 1, it means that the engineer doesn't have to be an expert to use this strategy (as for *in free prose*). On the contrary, to use the Goal structure driven strategy will requires a high level of expertise as the goal has to be formalized in details, so the value is 3. The last indicators have a qualitative value equal either at Low, Medium or High. Note that the Template driven strategy has two occurrences on the CL Map. However, it appears only one time in this table as it is the same strategy (values are the same).

Sections indicators. Indicators of a section are a combination of (a) the indicators relevant to its strategy, (b) the indicators relevant to its target intention, (c) the dynamic indicators (i.e. *Goal achievement degree*; if the section corresponds to a cycle, this indicator has a low value, if it is not, it has a high value), and (d) the indicators only relevant to sections (i.e. *Duration*). The *Duration* values are expressed in minutes and are impacted by the *Expertise degree* value of the section. For instance, the section S1 will be realized in 15 minutes if the *Expertise degree* is equal to 1. If this indicator has a bigger value, the length of time used to realize the section will be shorter. This means also that the *Duration* indicator will only have an interest for the Section concept.

Table 3. Indicator's values

	Expert role	Expertise degree	Formality degree	Goal Achievement degree	Relationship degree	Complexity degree	Duration
Intention							
Elicit a Goal	Analyst						
Write a Scenario	Analyst						
Conceptualize a Scenario	Analyst						
Strategy							
Initial Goal identification		1					
Template Driven			2				
Goal structure driven		3	3				
Linguistic			2				
Free prose		1	1				
Manual		3	1				
Computer supported		2	1				
Alternative discovery					Medium	Medium	
Composition discovery					Medium	Medium	
Refinement discovery					Medium	Medium	
Completness			1		Low	Low	
Section							
S1	Analyst	1					15 mn
S2	Analyst	3	3	Low			10 mn
S3	Analyst	2	3	Low			10 mn
S4	Analyst		2	Low			15 mn
S5	Analyst	3	3	High			10 mn
S6	Analyst	1	1	High			5 mn
S7	Analyst	2	1	High			10 mn
S8	Analyst	1	3	High			15 mn
S9	Analyst			High	Medium	Medium	20 mn
S10	Analyst			High	Medium	Medium	20 mn
S11	Analyst			High	Medium	Medium	20 mn
S12	Analyst		1	High			5 mn

4 MIG Process: Application of the Process Line Configuration

4.1 MIG Principles

As mentioned above, the MAP is an intentional process model. This formalism allows specifying process models in a flexible way by focusing on the process intentions, and on the various ways to achieve each of these intentions. Therefore, it has a teleological nature (it takes into account the teleological behavior of a process execution). It describes the intentions (goals, objectives) associated to a result that the designer wants to achieve [18] [19]. In this way, the MAP model presupposes decisions which concern intentions, strategies or elementary actions. We foresee the MAP guidance problem as a decision-making (DM) problem.

A DM problem is defined by the presence of alternatives. The concept of alternative designates the decision object. Any decision involves at least two alternatives that must be well identified. Alternatives are compared between them according to one or more criteria. Based on this, DM methods can be monocriterion or multicriteria. Using a single criterion is most used but it is not sufficient when the consequences of the alternatives to be analyzed are important [20]. Multicriteria DM methods, in contrast to a monocriterion approach, allow a more in-depth analysis of the problem because they consider various aspects. These methods deal with indicators having different nature (quantitative or qualitative). However, they are more complicated as they must be aggregated into a general value or function [20] [21]. From a DM viewpoint, the

Map guidance is characterized as follows: alternatives are intentions, strategies or actions; criteria are indicators.

4.2 MIG Approaches

There are two main steps in the MIG process: The customization of the process line and its runtime application. This process is described as a Map in Fig.3.

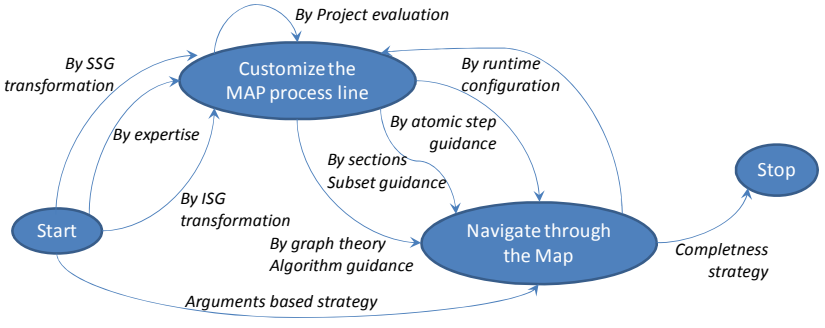


Fig. 3. MIG Process

Customize the MAP process line. To customize the MAP process line, the method engineer firstly defines MAP indicators values. He can use three ways: either by his expertise, or by transforming the ISG or SSG arguments. Secondly, he evaluates the project within this typology in order to define values available for this project.

Navigate through the MAP. The second step is to use these values for the guidance. The indicator typology may be used on a Map guidance following three mains approaches. The first one uses the indicator(s) to simplify the Map by suppressing some sections (*by section subset guidance*). The second one helps to select, right from the start of the navigation, between all the possible paths of the Map (*By graph theory algorithm guidance*). The last one guides each of the engineer steps, one by one - which is the initial and usual way to use a map (*By atomic step guidance*). Moreover, for each of these three kinds of guidance, there is also two separate ways to proceed which are the use of a single indicator or of multiple indicators. These approaches are described below and illustrated by examples using only quantitative indicators. The use of qualitative indicators is possible by applying appropriated DM methods.

The MIG process allows a backward step from the navigation to the customization as (a) the dynamic indicators have to be reevaluated after each section execution and (b) the navigation through the Map changes the context and it implies that the engineer may have to reevaluate the indicators values (*By runtime configuration*).

The engineer has the possibility to navigate through the Map without using the configuration but only the predefined SSG and ISG arguments (*Arguments based strategy*), which is the usual way to use a Map.

4.2.1 Sections Subset Guidance

This approach is applied when the goal is to eliminate the sections, which are not appropriated for the given indicators values.

Sections Subset by Single indicator. It is applied when only one indicator is available. The engineer has to define the needed value of this indicator in order to select an adapted sections subset. If, for instance, the engineer is a beginner, the *Expertise Degree* indicator must have a value equal to 1 (low). Using this approach allows to delete all the sections which are too difficult for the user. The sections subset of the CL Map is then composed of the following sections: S1, S4, S6, S8, S9, S10, S11, and S12. An engineer with a low level of expertise may execute all these sections.

Sections Subset by Multiple Indicators. This guidance has the same goal as the previous one but uses an aggregated value of several indicators following an MC method. For instance, weighting methods [22], outranking methods [20] or MAUT [21] aggregates criteria into a general value to be used as a single criterion. Several functions may be applied (sum, maximum, minimum, average, weighted sum.). For instance, the engineer selects the *Expertise degree* and the *Formality degree* as the two more important indicators and he wants to minimize their values. As aggregation rule, the engineer chooses to calculate the average value. The indicators do not have a value for all the sections. In order to obtain an aggregated value, we consider that a null value is equal to the lowest value of the indicator (see italic values on the Table 4). For instance, if a section does not specify the value of the *Expertise Degree*, we assume that any engineer may execute it and we affect the value 1 to this indicator. Once the aggregated values have been defined, the engineer defines the maximal average value that he could accept (here 1.5). The final sections subset includes: S1, S4, S6, S7, S9, S10, S11, and S12.

Table 4. Aggregated value of *Expertise degree* and *Formality degree* on the CL Map

Ind. Vs Sections	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**
Formality degree	*1*	3	3	*1*	3	1	1	3	*1*	*1*	*1*	1
Expertise degree	1	3	2	*1*	3	1	2	1	*1*	*1*	*1*	*1*
Aggregated value	**1**	**3**	**2,5**	**1**	**3**	**1**	**1,5**	**2**	**1**	**1**	**1**	**1**

It is interesting to note that the only difference between this subset and the subset from the previous section (4.2.1) is the section S8 replaced by the section S7 between the intentions *Write a Scenario* and *Conceptualize a Scenario*. The adjunction of the *Formality degree* to the *Expertise degree* has only this little impact.

4.2.2 Graph Theory Algorithms Guidance

This approach use Graph theory algorithms to find an optimal path through the Map.

Graph Theory Algorithms by Single Indicator. [23] shows the possible transformation of a map into a graph in order to use graph algorithms. Possible algorithms to apply are the *Shortest path*, the *Hamiltonian path* (path navigating through all the sections of the Map) and so on. The engineer selects an indicator, identifies all possible paths, and calculates values for each path (for quantitative indicators this is a sum and for qualitative indicators it can be a given value from the predefined set as [ENUM]).

For instance, the engineer decides to select a path, which minimizes time for executing the CL Map. He applies the graph theory algorithm for searching the Shortest path following the indicator *Duration*. The *Duration* values for all sections of the map

are indicated on the Table 5. Recall that in this example, the *Expertise degree* indicator impacts the *Duration* (an engineer with a low level of expertise will execute the section in more time than if he was an expert). The *Duration* values are given in the case when the *Expertise degree* is low. The possible paths set may be quite big as the Map contains cycles, iterations and back paths. To simplify our example, only the four shortest paths (which do not contain any cycle or iteration) are retained:

- Path 1 : {S1, S5, S7, S12} = 15 mn+10 mn +10 mn +5 mn = 40 mn
- Path 2 : {S1, S5, S8, S12} = 15 mn +10 mn +15 mn +5 mn = 45 mn
- Path 3 : {S1, S6, S7, S12} = 15 mn +5 mn +10 mn +5 mn = 35 mn
- Path 4 : {S1, S6, S8, S12} = 15 mn +5 mn +15 mn +5 mn = 40 mn

The shortest path is the Path 3 as it is the one which will be the quicker to execute.

Graph theory algorithms by Multiple Indicators. Graph theory algorithms may also be used with multiple indicators. Instead of using only one to select the better path, several indicators values are aggregated together in order to obtain a single value for each section. Then, they are used to select the optimal Map path. Several functions may be applied (sum, max, min, average, weighted sum) for aggregating indicators values.

In our example, the engineer selects three indicators: *Duration*, *Formality degree*, and *Expertise degree*, which are quantitative and have different measure scales. In order to obtain compatible scales for comparing them, normalization must be applied. In this case, the normalization is calculated as a percentage of each indicator maximal value. *Duration* values [5; 20], *Formality degree*, and *Expertise degree* values (ENUM:1, 2, 3) are reduced to the same scale [0; 1]. The normalized values are presented in the Table 5.

Table 5. Normalized values of *Duration, Expertise degree and Formality degree*

Ind. Vs Sections	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**
Duration	15	10	10	15	10	5	10	15	20	20	20	5
Formality degree	1	3	3	1	3	1	1	3	1	1	1	1
Expertise degree	1	3	2	1	3	1	2	1	1	1	1	1
Normalized values												
Duration degree	0,75	0,5	0,75	0,75	0,5	0,25	0,5	0,75	1	1	1	0,25
Formality degree	0,33	1	1	0,33	1	0,33	0,33	1	0,33	0,33	0,33	0,33
Expertise degree	0,33	1	0,67	0,33	1	0,33	0,67	0,33	0,33	0,33	0,33	0,33
Aggregated Value	**1,41**	**2,5**	**2,42**	**1,41**	**2,5**	**0,91**	**1,5**	**2,08**	**1,66**	**1,66**	**1,66**	**0,91**

The engineer wants to minimize duration, formality and expertise degrees, so he chooses the shortest path. The function used on the aggregated values is the sum.

- Path 1 : {S1, S5, S7, S12} = 1,41 + 2,5 + 1,5 + 0,91 = 6,32
- Path 2 : {S1, S5, S8, S12} = 1,41 + 2,5 + 2,08 + 0,91 = 6,90
- Path 3 : {S1, S6, S7, S12} = 1,41 + 0,91 + 1,5 + 0,91 = 4,73
- Path 4 : {S1, S6, S8, S12} = 1,41 + 0,91 + 2,08 + 0,91 = 5,31

The shortest path taking into account the three selected indicators is the Path 3.

4.2.3 Atomic Step Guidance

The usual way to use a map is to dynamically choose the sections to execute one by one.

Atomic step by Single Indicator. The engineer may use a specific indicator particularly relevant to its project, to help him to make his choice. First step is to select an indicator; the second is to determine the preference rule about it.

For instance, the navigation through the Map leads the engineer to execute the section S3 (<Elicit a Goal, Template driven strategy, Elicit a Goal>). Five possibilities are offered to the engineer to continue. He may execute one of the three sections which have the same target intention (S2, S3, S4) or go further in the Map to the intention *Write a Scenario* with the execution of the sections S5 or S6. If its predefined indicator is the *Duration* with a low value preference, this guidance will propose him section S6.

Atomic step by Multiple Indicators. The engineer may also use an aggregated value of several indicators (aggregation of the alternatives evaluations into a unique indicator, following a specific MC method). In this guidance, first step is to select several indicators; second to aggregate values; third to determine the preference rule about the results.

Within the same example as in the preceding section, the navigation has led the engineer to execute section S3. He can now execute S2, S3, S4, S5 and S6. He chooses three indicators: *Duration, Formality degree* and *Expertise degree* with a preference to a lowest final value. The normalized and aggregated values are described in Table 5. Between these six sections, the one that has the lowest value is the section S4.

5 Conclusion

We suggest considering Process models having a high level of variability as Process Lines. The number of variation points in processes increases the need for a better guidance. The proposed MIG approach, which is based on the indicators typology, enhances the guidance of the MAP process model by process lines configuration. In this particular process line, three main approaches are proposed to use the typology, (i) to pre-select some sections in the Map, (ii) to use Graph theory algorithms to select a specific path and (iii) to use a step by step guidance. Even if this approach has been created for the MAP model, it is applicable to any process model containing variation points.

Our future work will be twofold: a) to validate this typology on several well known processes modeled with Maps in order to define precisely in which situations the approaches may be used and b) to improve the Map Editor tool [24] and combine it with the Map executor tool currently on development. This tool will allow to draw Maps and configure them and to use this configuration in a computer-aided guidance of Maps.

References

1. Rolland, C.: Conceptual modeling in Information systems engineering. In: Capturing system intentionality with maps (2007)
2. Van Gurp, J.: Variability in Software Systems, the key to Software Reuse, Licentiate Thesis, University of Groningen, Sweden (2000)
3. Pohl, K., Böckle, G., van der Linden, F.: Software product line engineering: foundations, principles and techniques. Springer, Heidelberg (2005)
4. Svanberg: On the notion of variability in software product lines, working. In: IEEE/IFIP conference on software architecture (2001)
5. Bosch, J., et al.: Variability issues in Software Product lines. In: 4th international workshop on Product Family engineering (PEE-4), Bilbao, Spain (2001)

6. Bachmann, F., Bass, L.: Managing variability in software architectures. In: Proceedings of the 2001 Symposium on Software Reusability (SSR 2001), pp. 126–132. ACM Press, New York (2001)
7. Halmans, J.: Communicating the variability of a software product family to customers. Software and system modeling. Springer, Heidelberg (2003)
8. La Rosa, M., Dumas, M.: Configurable Process Models: How To Adopt Standard Practices In Your Own Way? BPTrends Newsletter (November 4, 2008)
9. Anón, A.I., Potts, C.: The use of goals to surface requirements for evolving systems. In: Proceedings of the 20th International Conference on Software Engineering (1998)
10. Yu, E.S.K., Mylopoulos, J.: Understanding "why" in software process modelling, analysis, and design. In: Proceedings of ICSE 1994, pp. 159–168 (1994)
11. Rolland, C., Prakash, N., Benjamen, A.: A Multi-Model View of Process Modelling. Requirements Engineering 4(4) (1999)
12. Rolland, C., Prakash, N.: On the Adequate Modeling of Business Process Families. In: BPMDS 2007 in conjunction with CAiSE 2007, Norway (2007)
13. Deneckere, R., Kornyshova, E., Rolland, C.: Enhancing the Guidance of the Intentional Model "MAP": Graph Theory Application. In: RCIS 2009, Fes, Morocco (2009)
14. Prakash, N., Rolland, C.: Systems Design for requirements expressed as a map. In: Proc. of the conference IRMA 2006, Washington, DC (2006)
15. Kornyshova, E., Deneckère, R., Salinesi, C.: Method Chunks Selection by Multicriteria Techniques: an Extension of the Assembly-based Approach. In: ME 2007, Geneva, Switzerland (2007)
16. Dietz, J.L.G.: Basic Notions Regarding Business Processes and Supporting Information Systems. In: Proceedings of BPMDS 2004, CAISE 2004 Workshops Proceedings, Latvia, Riga (2004)
17. Ralyté, J., Rolland, C., Plihon, V.: Method Enhancement with Scenario Based Techniques. In: Jarke, M., Oberweis, A. (eds.) CAiSE 1999. LNCS, vol. 1626, pp. 103–118. Springer, Heidelberg (1999)
18. Veblen, T.: Why is Economics not an Evolutionary Science? The Quarterly. Journal of Economics 12(4), 373–397 (1898)
19. Taylor, C.: The Explanation of Behaviour. Routledge, London (1964)
20. Roy, B.: Multicriteria Methodology for Decision Aiding. Kluwer Academic Publishers, Dordrecht (1996)
21. Keeney, R.L., Raiffa, H.: Decisions with Multiple Objectives: Preferences and Value Trade-Offs. Cambridge University Press, Cambridge (1993)
22. Keeney, R.L.: Foundations for Making Smart Decisions, IIE Solutions 31(5) (1999)
23. Deneckere, R., Kornyshova, E., Rolland, C.: Enhancing the Guidance of the Intentional Model "MAP": Graph Theory Application. In: RCIS 2009, Fes, Morocco (2009)
24. Thevenet, L.H.: Map Editor Tool for Intentional Strategic Alignment, Internal report (2009)

Appendix A: Authors

In order to access to *up to date* information about authors, please scan the following codes.

Rébecca Deneckère Elena Kornyshova

For scanning abilities, you may install the following barcode scanner software on your mobile: http://www.lynkware.com/support_devices.php

Appendix B: Indicators		Value domain		Connections			
		Quantitative	Qualitative	Map	Int	Str	Sect
Organisational Facet	Management commitment degree	3-grade scale	ENUM:{low, normal, high}	X	X		X
	Importance degree	3-grade scale	ENUM:{low, normal, high}	X	X		X
	Impact degree	3-grade scale	ENUM:{low, normal, high}	X	X		X
	Time pressure degree	3-grade scale	ENUM:{low, normal, high}	X		X	X
	Shortage of resources degree	3-grade scale	ENUM:{low, normal, high}	X			X
	Level of innovation degree	3-grade scale	ENUM:{low, normal, high}	X	X	X	X
	Size	3-grade scale	ENUM:{low, normal, high}	X			X
	Cost	REAL	ENUM:{low, normal, high}	X			X
	Nature of limited resources		ENUM:{financial, human, temporal, informational }	X	X		X
	Innovation nature		ENUM:{business innovation, technology innovation}	X	X	X	X
	Duration	REAL		X			X
Human Facet	Resistance degree	3-grade scale	ENUM:{low, normal, high}	X	X	X	X
	Conflict degree	3-grade scale	ENUM:{low, normal, high}	X	X		X
	Expertise degree	3-grade scale	ENUM:{low, normal, high}	X	X	X	X
	Clarity degree	3-grade scale	ENUM:{low, normal, high}	X	X		X
	Stability degree	3-grade scale	ENUM:{low, normal, high}	X	X		X
	Expert role		ENUM:{tester, developer, designer, analyst}	X	X		X
	User involvement		ENUM:{real, virtual}	X		X	X
	Stakeholder number	NUMBER		X		X	X
Application Domain facet	Formality degree	3-grade scale	ENUM:{low, normal, high}	X		X	X
	Relationships degree	3-grade scale	ENUM:{low, normal, high}	X		X	X
	Dependency degree	3-grade scale	ENUM:{low, normal, high}	X		X	X
	Complexity degree	3-grade scale	ENUM:{low, normal, high}	X		X	X
	Repetitiveness degree	3-grade scale	ENUM:{low, normal, high}	X		X	X
	Variability degree	3-grade scale	ENUM:{low, normal, high}	X	X	X	X
	Application type		ENUM:{intra-organization, inter-organization, organization-customer }	X	X		X
	Application technology		ENUM:{application to develop - includes a database, - is distributed, - includes a GUI}	X	X		X
	Dividing project		ENUM:{one single system, system-oriented subprojects, process-oriented subprojects, hybrid subprojects}	X	X	X	X
	Variable artefacts		ENUM:{organisational, human, application domain, development strategy}	X	X	X	X
Development Strategy Facet	Source system		ENUM:{code reuse, functional domain reuse, interface reuse}	X	X		X
	Project organization		ENUM:{standard, adapted}	X	X	X	X
	Development strategy		ENUM:{outsourcing, iterative, prototyping, phase-wise, tile-wise}	X		X	X
	Realization strategy		ENUM:{at once, incremental, concurrent, overlapping}	X		X	X
	Delivery strategy		ENUM:{at once, incremental, evolutionary}	X		X	X
	Tracing project		ENUM:{weak, strong}	X	X	X	X
	Goal number	NUMBER	ENUM:{one goal, multi-goals}	X			X

Author Index