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

LNBIP 81

Enterprise, Business-Process and Information Systems Modeling

12th International Conference, BPMDS 2011
and 16th International Conference, EMMSAD 2011
held at CAiSE 2011, London, UK, June 2011, Proceedings

Lecture Notes
in Business Information Processing 81

Series Editors

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

Enterprise, Business-Process and Information Systems Modeling

12th International Conference, BPMDS 2011
and 16th International Conference, EMMSAD 2011
held at CAiSE 2011, London, UK, June 20-21, 2011
Proceedings

Springer

Volume Editors

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

Selmin Nurcan
Université Paris 1 Panthéon Sorbonne, France
E-mail: nurcan@univ-paris1.fr

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

Pnina Soffer
Haifa University, Israel
E-mail: spnina@is.haifa.ac.il

Erik Proper
Public Research Centre Henri Tudor, Luxembourg
and Radboud University Nijmegen, The Netherlands
E-mail: erik.proper@tudor.lu

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

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

ISSN 1865-1348 e-ISSN 1865-1356
ISBN 978-3-642-21758-6 e-ISBN 978-3-642-21759-3
DOI 10.1007/978-3-642-21759-3
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011929237
ACM Computing Classification (1998): J.1, H.4.1, H.3.5, D.2

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This book contains the proceedings of two long-running events held in connection to the CAiSE conferences relating to the areas of enterprise, business-process and information systems modeling:

- The 12th International Conference on Business Process Modeling, Development and Support (BPMDS 2011)
- The 16th International Conference on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD 2011)

The two working conferences are introduced briefly below.

BPMDS 2011

BPMDS has been held as a series of workshops devoted to business process modeling, development and support since 1998.

During this period, business process analysis and design has been recognized as a central issue in the area of information systems (IS) engineering. The continued interest in these topics on behalf of the IS community is reflected by the success of the last BPMDS workshops and the recent emergence of new conferences and workshops devoted to the theme. Facing this trend, in 2011 BPMDS became a working conference.

While changing the status of BPMDS, we preserved the basic principles of the BPMDS series:

1. BPMDS serves as a meeting place for researchers and practitioners in the areas of business development and business applications (software) development.
2. The aim of the event is mainly discussions, rather than presentations.
3. Each event has a unique theme.

Previously, each workshop had a relatively narrow theme, different each year, related to the current progress in the area of BPM. Our intention when becoming a working conference was to solicit papers related to business process modeling, development and support (BPMDS) in general, using quality as a main selection criterion, instead of relevance to a narrower theme.

As a working conference, our aim was to attract more papers describing mature research, still giving place to industrial reports and visionary papers. We kept the principle of a defined theme for the event, but used relevance to the theme as a criterion only for visionary papers, describing innovative research ideas which are not yet completely mature. In addition, we suggested to the authors of research papers and experience reports, wherever possible, to make a connection to the theme.

The theme chosen for BPMDS 2011 visionary papers was 'Making BPM theory work in practice: "There is nothing more practical than a good theory (Kurt Lewin)".'

BPMDS 2011 received a record number of 61 submissions from 25 countries (Australia, Austria, Belgium, Brazil, Canada, Colombia, Denmark, France, Germany, India, Iran, Israel, Italy, Japan, Latvia, Luxembourg, The Netherlands, Norway, Portugal, Spain, Sweden, Uganda, United Arab Emirates, UK, USA). The management of paper submission and reviews was supported by the EasyChair conference system. Selecting the papers to be accepted was a worthwhile effort. Each paper received at least three reviews. Eventually, 22 high-quality papers were selected; among them three experience reports and two visionary papers.

The accepted papers cover a wide spectrum of issues related to business process development, modeling, and support. They are organized under the following section headings:

- BPMDS in Practice
- Business Process Improvement
- Business Process Flexibility
- Declarative Process Models
- Variety of Modeling Paradigms
- Business Process Modeling and Support Systems Development
- Interoperability and Mobility

We wish to thank all the people who submitted papers to BPMDS 2011 for having shared their work with us, as well as the members of the BPMDS 2011 Program Committee, who made a remarkable effort reviewing the large number of submissions. We also thank the organizers of CAiSE 2011 for their help with the organization of the event, and IFIP WG8.1 for the support.

The goals, format, and history of BPMDS can be found on the website: http://www.bpmds.org

April 2011

Ilia Bider
Selmin Nurcan
Rainer Schmidt
Pnina Soffer

EMMSAD 2011

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 toward standardization (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 workshop 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 2011 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 2011 was the 16th in a very successful series of EMMSAD events, previously held in Heraklion, Barcelona, Pisa, Heidelberg, Stockholm, Interlaken, Toronto, Velden, Riga, Porto, Luxembourg, Trondheim, Montpellier, Amsterdam and Hammamet. This year we had 31 submissions by authors from 16 different countries (Australia, Austria, Belgium, Brazil, France, Germany, Israel, Italy, Luxembourg, Malaysia, The Netherlands, Norway, Spain, Sweden, the UK, and the USA). The management of paper submission and reviews was facilitated by use of the EasyChair conference system. After an extensive review process by a distinguished international Program Committee, with each paper receiving at least three reviews, we accepted the 16 papers that appear in these proceedings. Congratulations to the successful authors!

Apart from the contribution by paper 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. We are also grateful for the sponsorship of the European INTEROP Network of Excellence, AIS-SIGSAND, the Enterprise Engineering Network, and the ORM Foundation. For more information on past and future EMMSAD conferences, please see our website http://www.emmsad.org

April 2011

Terry Halpin
John Krogstie
Erik Proper

Organization

BPMDS 2011 Organizing Committee

Selmin Nurcan	University Paris 1 Panthéon Sorbonne, France
Pnina Soffer	University of Haifa, Israel
Rainer Schmidt	Aalen University, Germany
Ilia Bider	IbisSoft, Stockholm, Sweden

BPMDS 2011 Industrial Advisory Board

Ilia Bider	IbisSoft, Stockholm, Sweden
Gil Regev	EPFL and Itecor, Switzerland
Lars Taxén	Linköping University, Sweden

BPMDS 2011 Program Committee

Sebastian Adam	Fraunhofer IESE, Kaiserslautern, Germany
Antonia Albani	Delft University of Technology, The Netherlands
Eric Andonoff	IRIT, Université Toulouse 1, France
Ilia Bider	IbisSoft, Stockholm, Sweden
Claude Godart	LORIA, Nancy-Université, France
Stewart Green	University of the West of England, UK
Giancarlo Guizzardi	Federal University of Espírito Santo (UFES), Brazil
Paul Johannesson	Royal University of Technology, Stockholm, Sweden
Udo Kannengiesser	NICTA, Australia
Christian Koot	Aalen University, Germany
Agnes Koschmider	Karlsruhe Institute of Technology, Germany
Marite Kirikova	Riga Technical University, Latvia
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
Oscar Pastor	Université Polytechnique de Valencia, Spain
Louis-Francois 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	Aalen University, Germany
Pnina Soffer	University of Haifa, Israel
Markus Strohmaier	University of Toronto, Canada
Lars Taxén	Linköping University, Sweden
Roland Ukor	FirstLinq Limited, UK
Barbara Weber	University of Innsbruck, Austria
Jelena Zdravkovic	Royal University of Technology, Stockholm, Sweden
Michael zur Muehlen	Stevens Institute of Technology, USA

BPMDS 2011 Additional Reviewers

Ian Beeson	Jens Kolb
Xavier Burgués	Vera Kuenzle
Dolors Costal	Kevin Lee
Michael Eisenbarth	Maria Leitner
Walid Fdhila	Juergen Mangler
Cristina Gómez	Jens Ohlsson
Anne Groß	Jin Sa
Sonja Kabicher	Sherif Sakr
David Knuplesch	Robert Scott
Thomas Kohlborn	Uldis Sukovskis

EMMSAD Steering Committee

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

Conference Co-chairs

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

EMMSAD 2011 Program Committee

Stephan Aier	University of St. Gallen, Switzerland
Antonia Albani	Delft University of Technology, The Netherlands
Herman Balsters	University of Groningen, The Netherlands
Annie Becker	Florida Institute of Technology, USA
Giuseppe Berio	University of Turin, Italy
Linda Bird	National E-Health Transition Authority, Australia
Peter Bollen	Maastricht University, The Netherlands
Nacer Boudjlida	Loria, France
Sjaak Brinkkemper	Utrecht University, The Netherlands
Andy Carver	INTI International University, Malaysia
Olga De Troyer	Vrije UniversiteitBrussel, Belgium
David Embley	Brigham Young University, USA
Mathias Ekstedt	KTH - Royal Institute of Technology, Sweden
John Erickson	University of Nebraska-Omaha, USA
Gordon Everest	University of Minnesota, USA
Peter Fettke	Institute for Information Systems (IWi) at the DFKI, Germany
Ulrich Frank	University of Duisberg-Essen, Germany
Andrew Gemino	Simon Fraser University, Canada
Remigijus Gustas	Karlstad University, Sweden
Wolfgang Hesse	University of Marburg, Germany
Stijn Hoppenbrouwers	Radboud University Nijmegen, The Netherlands
Mike Jackson	Birmingham City University, UK
Paul Johannesson	Stockholm University, Sweden
Pontus Johnson	KTH - Royal Institute of Technology, Sweden
Pericles Loucopoulos	Loughborough University, UK
Kalle Lyytinen	Case Western Reserve University, USA
Florian Matthes	Technical University of Munich, Germany
Raimundas Matulevičius	University of Tartu, Estonia
Graham McLeod	University of Cape Town, South Africa
Jan Mendling	Humboldt University, Berlin, Germany
Wolfgang Molnar	Public Research Centre Henri Tudor, Luxembourg
Tony Morgan	INTI International University, Malaysia
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
Colette Rolland	University of Paris 1, France
Kurt Sandkuhl	Jönköping University, Sweden

Peretz Shoval	Ben-Gurion University of the Negev, Israel
Guttorm Sindre	University of Trondheim, Norway
Il Yeol Song	Drexel University, USA
Carson Woo	University of British Columbia, USA
Martin Zelm	CIMOSA, Germany

Additional Reviewers for EMMSAD 2011

Alexis Aubry
Thomas Buechner
Mikael Berndtsson
Khaled Gaaloul
David Gouyon
David Heise
Hannes Holm
Constantin Houy
Heiko Kattenstroth
Ritu Khare
Per Närman
Pia Närman
Waldo Rocha Flores
Sascha Roth
Xuning Tang
Ornsiri Thongoom
Armella-Lucia Vella
Jürgen Walter
Ilona Wilmont

Table of Contents

BPMDS in Practice

Business Process Improvement

Business Process Flexibility

Declarative Process Models

Variety of Modeling Paradigms

Business Process Modeling and Support Systems Development

Interoperability and Mobility

EMMSAD 2011

Workflow and Process Modeling Extensions

Requirements Analysis and Information Systems Development

Requirements Evolution and Information System Evolution

New Approaches for Systems Engineering and Method Engineering

Data Modeling Languages and Business Rules

Variability within Software Product Line Engineering

Conceptual Modeling Practice

Enterprise Architecture

Business Process Management for Open E-Services in Local Government - Experience Report*

Petia Wohed[1], David Truffet[2,**], and Gustaf Juell-Skielse[1]

[1] Stockholm University, Sweden
{petia,gjs}@dsv.su.se
[2] Esri, Australia
dtruffet@esriaustralia.com.au

Abstract. E-government has become one of the most prominent means to reform the public sector. Building e-government embraces a variety of efforts both at a centralised level, (e.g. the integration of and communication between systems across different agencies, domains and geographies), and at local levels such as the development of e-services for the provision of 24/7 public sector agencies. In this paper, we report on the results of a project aimed to develop e-services as a part of the e-government initiative in Sweden. The project was carried out at the elderly and handicapped unit at one municipality. The e-services considered in the project were also intended to open up the underlying social services and are, therefore, referred to as open e-services. We discuss the results of the development of one such e-service as a proof-of concept solution for which a business process management system is used. We present the solution and explain the features of using a business process management system as a back-end system.

Keywords: Business process management, e-services, e-government.

1 Open E-Services in Local Government

Local governments in Sweden are responsible for a large share of public services in the fields of education, care and healthcare. Elderly care and care for the disabled are two important tasks and account for almost 30% of municipal budgets. To meet the challenges of an aging population [5], local governments are strengthening their capabilities in the area of assisted living services [6]. Focus is placed on streamlining the administration associated with the services, on opening up the services, and on developing e-services.

Opening up the services includes the removal of the formal decisions of granting or rejecting services to citizens taken by municipal officials. For instance 26

* This work is funded in part through the project Open Social Services financed by The Swedish Agency for Research and Innovation for Sustainable Growth (Vinnova).
** Work conducted during a visit at the Queensland University of Technology.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 1–15, 2011.

(out of 290) municipalities in the country provide the emergency phone service as an open service [7, p. 28], i.e. the service is ordered and subscribed to by citizens instead of applied for and then formally approved by authorities.

The *development of e-services* has initially resulted in the development of application forms that can be submitted electronically. Although these *application e-forms* imply some simplification for the citizen they offer only a limited improvement for those who handle the applications at the government. To develop e-services that support the entire processes, we engaged in a project aiming at the development of three *open e-services* for the elderly and disabled care at a municipality. Our role in the project included the business process analysis and design for the e-services, as well as an analysis of the use of open source software for their implementation[1]. To provide support for the entire processes, we used a business process management system (BPMS) as a back-end system. To analyse the use of open source software, we prototyped the first e-service, i.e. the emergency phone application e-service, in an open source BPMS called YAWL[2]. This paper reports on the prototype as a proof-of-concept implementation[3].

The paper is structured as follows. Section 2 presents the steps we followed to develop the e-service. Section 3 outlines a number of goals considered in the provision of e-services. Section 4 presents the solution. Section 5 and an Epilogue conclude the work with an analysis of how well the BPMS-based solution meets the goals and reflections from the work.

2 Method of Work

To develop the open social e-service, we progressed through the following phases: process analysis, process specification and e-form development. During the *process analysis* phase, an as-is analysis of the current process was performed. For this, a number of workshops were carried out with a work-group consisting of six municipal officials, two executives, a representative from the IT department, and sometimes a representative from the social service provider, who is a sub-supplier for the municipality.

The purpose of the workshops was to gather information about the existing routines. The workshops were complemented with four in-depth interviews with representatives from the different roles during which also the current IT system was looked into. After every workshop the results were documented in YAWL process models and validated during the next following workshop. The process models were also periodically presented to municipality management.

During the *process specification* phase, a to-be process (solving the problems identified during the process analysis) was designed and agreed upon. The work was carried out in a similar way as before, i.e. through workshops with officials

[1] The actual implementation of the services is the responsibility of two software development companies.

[2] www.yawlfoundation.org

[3] The prototype is available for download at http://dash.dsv.su.se/2010/10/19/sundsvall42/

from the municipality. Management decided on a set of decision criteria for the open social service. The design was presented at management level and after agreed upon prototyped in YAWL. Before the work on prototyping was initiated, the goals with the prototype were listed. Some of them related to limitations with the current way of working, whereas others reflected the general ambitions for e-services and use of integration technologies.

During the *e-form development* phase, the e-service's interfaces with its users were designed. The work was performed in workshops with the same group of officials as before and validated by the executives. The to-be process and the municipal web interface template and guidelines were followed. When ready the forms were implemented in the prototype.

3 Considerations for e-Services

When developing e-services we need to distinguish between three types of goals that need to be achieved within the legislative constraints applied by the EU and the Swedish state: goals associated with the citizen, goals of the municipality (granting and subsidizing the provision of the services) and goals associated with the service providers (i.e. the companies carrying out the services).

The goals related to citizens include:

G1 *Increased access to social services* with respect to time, geographic location and accessibility. Time refers to 24/7 government, geographic location refers to the possibility of applying for a service from home and accessibility is a legislative requirement [1] of the authorities investing in e-government efforts. Since the services are intended for the elderly and infirm, it is important that these groups of citizens can, if needed, access them via assistance devices such as audio and brail readers.

G2 *Increased openness* by providing a sense of control to the citizens:

a) Before submitting an application, inform the citizen about what is available, what is expected from them, and what they can expect from the authority. In particular, *make explicit the decision criteria* used for evaluating their application.
b) Once an application has been submitted, communicate status changes to the citizen so they know where they are in the process. Provide the citizen with the ability to intervene in the process, e.g. withdraw their application.
c) Once application handling has ended, notify the citizen of the outcome and provide them with the capability to review the result at a later date and/or challenge the result.

The goal related to the municipality is to develop a solution with the following properties:

G3 *Business orientation* The e-services should support the administration of citizen applications for social services. The technology should be used to

streamline the business process so that the lead time from application receipt to service delivery decreases and when motivated handling time spent on individual applications is reduced.

G4 *Visibility* Officials at different levels should be able to continually monitor individuals as well as sets of citizens and their statuses. In addition, the solution should fulfil both real time and historical reporting requirements.

G5 *Cost effectiveness* The solution should provide a low overall cost for a large number of e-services. This cost includes implementation, licensing, operation and maintenance.

G6 *Flexibility* The solution should be able to adapt and suit evolving requirements. This means that significant changes in a business process should have little impact on the system. Flexibility entails low cost of change.

G7 *Scalability* implies a reducing cost to the development of new services. Scalability lowers the average cost per service as the number of services implemented in the system increases. In our case, the e-service shall provide a general proof-of-concept solution applicable to other open e-services.

The goals related to the service providers (sub-suppliers to the municipality) of the social service include:

G8 *Increased integration with sub-suppliers* Currently, communication with sub-suppliers is manual and paper-based. Although information is already present in the municipal information system forms are still filled in by hand and faxed to the sub-suppliers. The same piece of information is filled in multiple times (in different forms). The communication routines with the sub-suppliers are fairly straightforward; however, using modern ICT would significantly reduce the time currently spent on shifting information around.

Legislative constraints applied by the EU and the Swedish state include:

G9 *Compliance with the law* The e-service dealing with citizen applications for a social service and as such providing a front-end for the service, must conform to the Social Services Act.

G10 *Multi-lingual support* A law requiring the support of national minorities and minority languages [3] came into operation in Sweden in January 2010. According to this law, if requested the municipalities must provide social services to a citizen of a given minority in their minority language. As the e-services will act as the first step of social services delivery, the multi-lingual requirement also applies to them. Furthermore, multi-lingual e-government is highly relevant in a wider European context.

G11 *Consideration of open source solutions* In "Strategy for Authorities' Work with e-Government", the Swedish government states that when procuring and developing software all authorities shall consider and evaluate open-source alternatives [4, p. 73].

4 The Solution

We aimed to develop an e-service to support the entire process of dealing with citizen applications for emergency phones. Since BPMSs are designed to support

business processes in organisations, they were our natural choice of software. To evaluate the use of open source solutions (goal G11) we used YAWL. YAWL was selected because it is one of the larger BPMSs distributed through an open source repository fulfilling our requirements on functionality [9,2]. However, the results of our work are general and another BPMS with similar functionality could have been used instead.

A basic principle for the development is to present users with a simple and intuitive interface without making them aware of the use of a BPMS as a back-end system. Therefore, the interface towards the citizens and users is realised through mail clients. The solution contains the following parts: (i) a process model with associated resource model; (ii) a set of custom forms used for displaying or acquiring data relevant for the process; (iii) an email notification demon for forwarding the allocated work to the corresponding mail clients; and (iv) case initiation by citizen through electronic application submission. In addition, (v) multi-lingual support was added to fulfill goal G10. Furthermore, the accessibility criteria defined for Europe were also considered (G1). The remainder of this section presents the solution.

4.1 The Emergency Phone Application Process

A YAWL model capturing the emergency phone application process is shown in Figure 1. Tasks are represented as rectangles. They may be manual (i.e. tasks requiring input from the user), notification tasks (i.e. notification emails sent to users with no actions required) or automated tasks. Tasks might have decorators and colours. The decorators are used to capture joins and splits of the control flow. There are three types of decorators - AND, XOR and OR decorators - with each coming in two variants, join and split. The colors are used to show the different roles responsible for the execution of the tasks. Lack of color indicates that a task is either automated or a routing task.

The process starts with the submission of an application by a *citizen*. Based on a number of predefined criteria the application is either routed to manual handling or progressed to emergency telephone equipment installation. In the

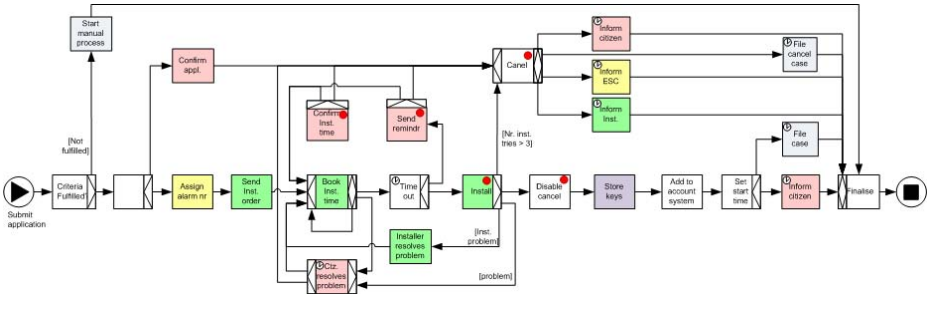


Fig. 1. The emergency phone application process

later case, an alarm number is obtained from the *emergency service centre*. Then, an installation order is sent to all *installers* and allocated to the first one to accept it. This installer is then responsible for the installation of the emergency telephone equipment in the citizen's home. The installer calls the citizen and schedules the installation. A confirmation of the time is sent to the citizen, and 24 hours before installation a reminder is sent and the process progresses to the installation task. At installation, installation data is obtained and keys from the citizen's home checked out. The keys are handed over to the *emergency group*, serving the emergency calls, who stores them securely. Henceforth, the citizen can start using their emergency phone. They are added to the accounting system and the start time for the service fee payment is calculated. The citizen is informed about the start service and payment time and the case is filed. Although electronic filing is legally approved in Sweden, filing at the municipality is still done manually by a *customer relationship officer*.

4.2 The Data and Custom Forms of the Process

In YAWL, all data is presented in XML. Working data is stored in process variables whose type is defined through the XML Schema language. A distinction is made between case and task variables. For each task the data needed for its execution or resulting from its execution is defined through input and output variables. These variables are associated with the case variables of the process. When a work item of a task is checked out, the engine populates its input data with the relevant values from the case variables. At completion, the engine transfers the values of its output parameters to the corresponding case variables. The associations are defined through the XQuery language. See Figure 2 for an example of input and output parameters for the task Install.

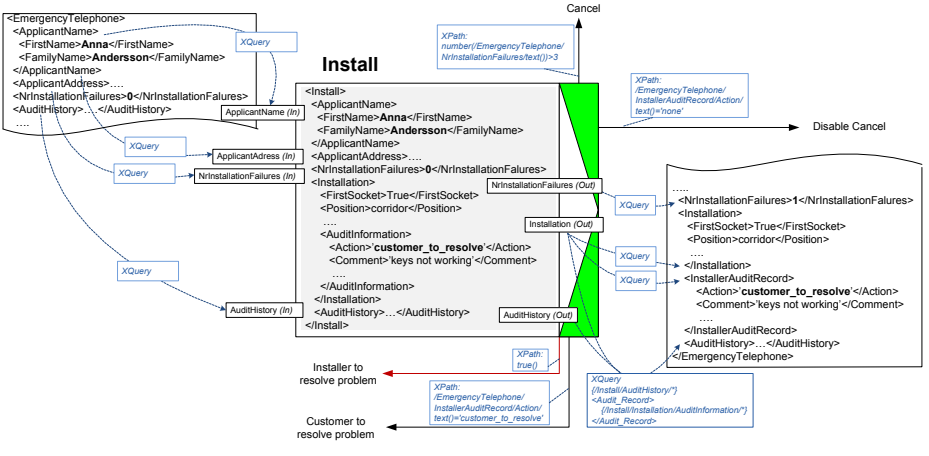


Fig. 2. Sample data for the task Install

Data-based routing decisions are made based on the values in the process variables. For this, different conditions are specified on the outgoing arcs of the XOR- and OR-split decorated tasks. The conditions are expressed in the XPath language. In the example in Figure 2, if an installation has failed four times (NrInstallationFailure > 3), the process will be interrupted.

Many of the tasks in the process are manual. For these tasks, data from the user is obtained through forms. To provide forms with the look and feel requested by the municipality, we developed custom-made web forms and used the form connector service in YAWL. Figure 3 shows such a form for the Install task.

The custom forms were developed using Java Server Pages (JSP). A JSP form was created for each manual task. A form contains: header and footer information; a definition of the variables that are input to or output from it; a number of includes that are HTML fragments for capturing and/or displaying the relevant variables; and a number of buttons/functions.

Each form uses interactive data validation against the schema definitions held within the process model. The forms obtain the XML schema type definitions from the process model and validate the supplied data against them. This implies that the data validation, which traditionally has been part of the application development, is now the exclusive concern of the process definition.

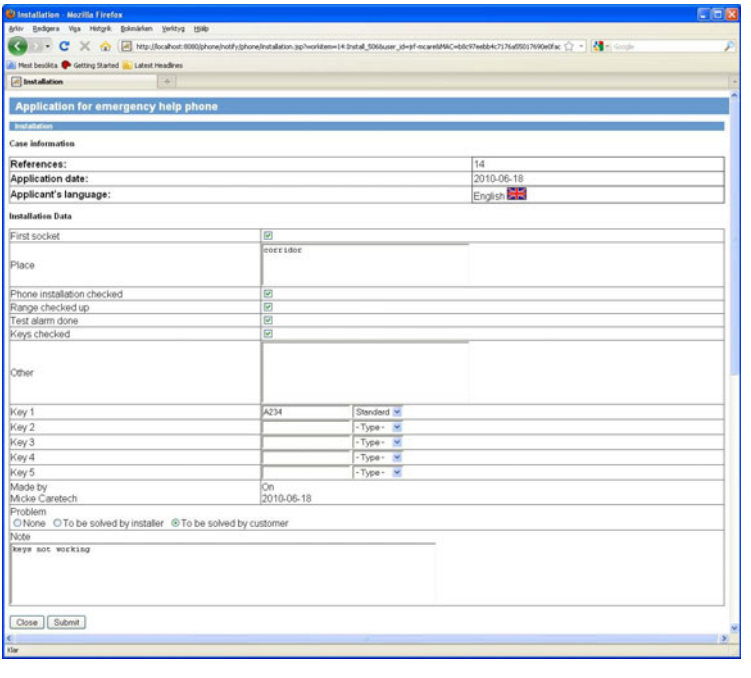


Fig. 3. Part of the form for the task Install

4.3 Email Notification Demon

The email notification demon periodically interrogates the work list of each user. When a work item appears as either offered or allocated the demon forwards it to the email client of the user. To compose the email, the notification demon opens the relevant web form, extracts and rewrites the HTML from it to a form compatible with email[4] and attaches the URL that would have been used by the YAWL engine. The corresponding work item is started when the user follows this URL. To maintain security the user is required to provide their credentials. In this way, an email account resembles a work queue in YAWL.

4.4 Case Initiation by Citizen

A part of the solution in YAWL has been to trigger the process through the receipt of an electronic application form. In Figure 1 the input condition "Submit Application" is used to visualize this. However, it should be noted that "Submit Application" is not considered as a real task by the YAWL engine. Traditionally, YAWL requires a case to be created by an administrator on behalf of the citizen before any tasks can be enabled. However, our solution provides the citizen with a *submission form* which upon completion launches a case. This form differs from the remainder of the custom forms in that it can be viewed by the users before an underlying case has been initiated. In addition, the submission form *launches a case* in contrast to the other custom forms that *complete work items*. The first page of the submission form is displayed in Figure 4. On submission of the form, the process is started and the user notified with a case reference number.

4.5 Multi-lingual Support

We developed a multi-lingual solution to support the Swedish and English languages (note that the forms in figures 4 and 3 are in Swedish and English, correspondingly). The solution is easily extensible to other languages such as Sami. Multi-lingual support is typically provided by duplicating all forms for the supported languages. Furthermore, many multi-lingual solutions provide parallel solutions, i.e. an English and a Swedish solution, which do not allow a transition between the different languages.

Our solution is more general and allows a variety of languages to be used within one case. Communication with the citizen is carried out in the language of their preference, which is determined by the language used in the submission form. However, throughout the process different users might use different languages. For instance, a case might be initiated in English and filed in Swedish[5].

[4] The rewriting includes the removal of all JavaScript, removal of the HTML form mark-up (text input fields, buttons, drop-down lists), and inlining/attaching images (e.g. logos etc.).

[5] A realistic scenario is the initiation of a case in Sami (which is one of the minority languages to be supported according to the law [3]) and filed in Swedish, which is the official language in the country.

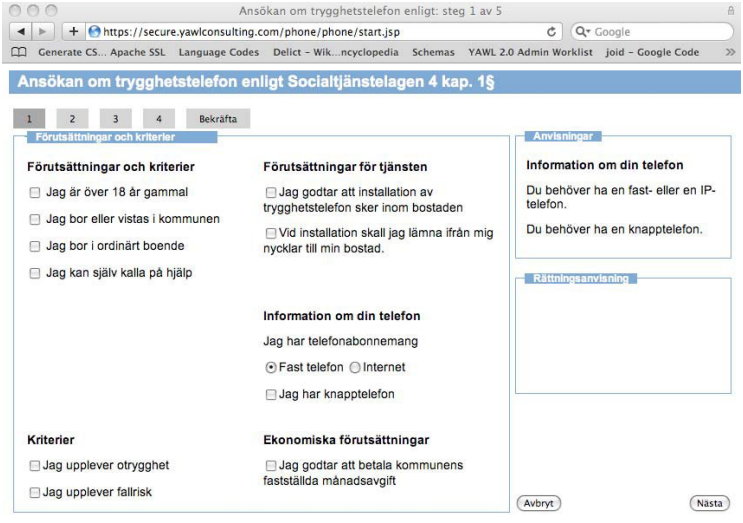


Fig. 4. The JSP form for the task 'Submit Application', in Swedish

The solution is based on two components: one mono-lingual solution and a translation filter. In addition, language capabilities are defined for the different users to reflect their language skills. The translation filter exploits the content of the forms and translates them via a translation file to the required language for the given user. A translation file contains each phrase from the mono-lingual solution and the corresponding phrase in the supported language. A new language is added by the addition of the corresponding translation file.

5 Discussion

In the previous section, we described the major points of our solution. Here, we discuss how the prototyped solution meets the goals outlined in Section 3. In the second part of the section, we reflect on the experience gained from the work. Some of these reflections can be used as input for the further development of YAWL or BPMSs in general, whereas others can be used as input to developers involved in projects similar to ours.

Fulfilment of the Goals

G1 Increased access to social services with respect to time, geographic location and accessibility. The use of e-service technology automatically increases the time and geographical accessibility to the service. To meet the requirements on accessibility, we designed the forms to be neutral to the client device accessing them. See, for example, Figure 5, where the custom form from Figure 4 is displayed in a text-based client. For this we followed the constraints on syntax and structure of web pages defined by W3C [8].

Fig. 5. The JSP form from Figure 4 displayed through a text-based client

G2 Increased openness To fulfil *G2a*, special attention was paid to providing adequate help to the citizen using the e-service. An introductory information page for the whole service was developed. The municipality layout guidelines, where the right column on the forms (Figure 4) is entirely dedicated to help messages and corrections, were strictly followed. The help text provided was context sensitive, i.e. when hovering with the mouse over the entry fields of a form, the corresponding help text would appear in the help area of the form. All texts and help messages were worded during the workshops with officials from the municipality. Furthermore, the decision criteria used for evaluation were defined by the executives and implemented in the web forms. In this way, the criteria were made explicit and the decisions transparent to the citizens.

To fulfil goals *G2b* and *G2c* we provide a case reference number after a citizen submits an application and notify them about all essential state changes during the process, for example, two notification tasks are introduced, one notifying the outcome of the service decision and one notifying equipment installation and service fee decision. In addition, the citizen is notified after an installer schedules an installation time with them. They are also notified in case any problem with the booking or installation arises. Furthermore, the citizen can withdraw their application at any time before installation. Finally, all cases that might be subject to rejection are redirected to manual handling and the challenging of a rejection is treated by the present manual routines.

Goals *G3*-*G7* are met automatically due to the fact that we used a BPMS as a back-end system. BPMSs coordinate activities, thereby ensuring that the

appropriate resource is notified about the work they need to perform and provided the required information needed for that work. BPMSs are configured through process models (*G3*). They provide visibility and transparency of the processes being managed and the cases present in the system. Through the implementation of business activity monitoring modules, they allow organisations to review a process as a whole or down to individual cases and support both real time and historical reporting requirements (*G4*).

Solutions implemented in a BPMS are generally meant to be low cost because a large number of processes (in this case, e-services) can be supported through the system (*G5*). However, high licensing costs and vendor lock-in can easily make investment in a BPMS a costly endeavour. Investment in an open source BPMS solution tends to entail the same hardware/infrastructure, design, implementation and operational costs as those for a proprietary system, but removes the licensing costs and, through the escape of vendor lock-in, can also reduce maintenance costs.

Using BPMSs implies separating business process logic from application logic. This means that changes in organisation processes can be easily reflected in a system by changing the corresponding process models (*G6*). At the same time, new business processes can be added by defining their process models in the system (*G7*). Moreover, removing business logic from the application development and placing it into process models allows a high degree of task reuse between a large number of processes. Furthermore, application development transfers to application component development, where separate components are implemented to define the behaviours of the automated tasks. With application development broken into component-per-(automated) task development, the application development effort is significantly reduced and sometimes even removed for the addition of new processes. In our case, this implies that as the number of services implemented in the system increases, the average cost of implementation per service decreases (*G7*).

To achieve *G8*, Increased integration with sub-suppliers, we integrated several "external" users into our solution. These are the citizen, the emergency group, the installer and the emergency service center (ESC). Parts of this integration imply changes and optimisation in the current process (*G3*), for example, the provision of alarm numbers from the ESC is today handled in bundles and the responsibility for allocating the numbers delegated to the emergency group. By providing the ESC direct access to the system, the number allocation routine is simplified. This direct access is enabled through the email notification demon - a mechanism that minimises external access and keeps the underlying system invisible. The integration of the citizen and the installer company is implemented in the same way. For the emergency group, which already has access to the municipal IT system, the integration work manifested differently. Currently, an essential part of the process data that is relevant to the emergency group is missing in the IT system (a large amount of the process documentation is paper-based and sent by fax). To solve this problem and achieve full integration, we focused on capturing and implementing the complete set of process data.

To achieve *G9, Compliance with the law*, the Department of Law at Lund University was consulted during the process analysis and specification phases. To achieve *G10*, a translation filter was implemented and a couple of translation files created (see Section 4.5). Finally, *G11* was addressed by prototyping in the open source system YAWL.

Reflections

Although the BPMS we used provided a number of advantages there were also some minor limitations. If these were removed, the development effort would further be simplified. We reflect on these here to provide input for the further development of YAWL.

Process initiation should be considered a first class citizen. In many cases, particularly in the area of e-government and e-commerce, there is a need to be able to launch a process through the submission of a form by a citizen or customer. We have implemented this as an extension to YAWL; however, we consider that this functionality should be managed by the BPMS. Implementing such functionality in YAWL would entail the following changes: the capability to assign roles to the input condition to specify which resource is allowed to start a process; and the possibility of assigning a form to the input condition for laying out the process input variables.

Allow the dynamic allocation of work based on capabilities. Currently, YAWL supports dynamic allocation of work based on case variables holding information about user id (i.e. deferred allocation). YAWL is also capable of allocating work based on capabilities, but these need to be defined during design time and cannot be changed on a per-case basis. Providing a dynamic resource allocation also for the capabilities would enable better targeting of task allocation. In our particular case, work would have been offered according to roles and language capabilities.

Provide a graphical presentation of process and task data. During the process analysis and specification phases we used YAWL's graphical process modelling language for all communication with business users. This was done despite the common attitude that a simplified language than the language of implementation should be used for communication with business users. When coming to the presentation of the control flow, our experience of using YAWL is positive and models similar to this in Figure 1 were successfully used. However, to present the data in the process we needed an additional, music line notation presenting a note line for each user with associated input and output data provided to/by them for every manual task. Because the data changed several times during the analysis, these changes needed to be reflected both in the YAWL model and in the side notation which was time-consuming and error prone. Graphical support for the data in the process editor would significantly simplify this work.

Support form design. YAWL provides automatic form generation, which allows rapid prototyping. We used this feature a couple of times to demonstrate intermediate prototypes to business users to validate our work. Our impression of

the rapid prototyping was generally positive. However, we faced the following shortcoming. In the automatically generated forms, all data fields are presented in a vertical sequence, the ordering of which cannot be controlled by the developer. After our process was validated and the data relevant to it agreed upon we continued with the interface design (i.e. the construction of the custom forms). During the work with the custom forms a number of changes to the data structure were again requested by the business users. The proper visualisation using the custom forms provided business users with a deeper understanding of the underlying process data. So, although functional for the developer, the automatically generated forms turned out not to be that useful for communication with the business users. We believe that the addition of a form designer to YAWL would be greatly beneficial for this purpose.

A general, non-YAWL-related reflection comes from our work with the multi-lingual goal. The provided multi-lingual solution, separating the language from the user interface layer, offers a couple of advantages: it allows the use of different languages during different phases of a process; and it facilitates changes in the form logic by only necessitating updates in one place. Nevertheless, the translation took longer than expected and desired. It was done out of context and fragmented. 'Out of context' means that phrases needed to be translated in isolation in a table in the translation file rather than on the form where they appeared. Fragmentation was caused by the fact that some of the phrases were split into multiple parts because they accommodated both static and dynamic content. To streamline the work, we developed a systematic approach and applied it to translating into a third language. However, even if the set-up time for adding a new language can be reduced, multi-lingual support is still a resource demanding activity.

Our final reflection relates to the multi-lingual set-up in which we worked. Because we used YAWL process models for communication with the business users, the models were documented in Swedish. The prototyping was, on the other hand, done in English, which is a rather usual scenario when outsourcing development work. As multi-lingual models are not supported by YAWL (nor, to our knowledge, by any other BPMS) two versions of the model needed to be defined and maintained: a Swedish and an English one. At the same time, because the use of a BPMS truly facilitated iterative development, the solution was changed multiple times, long after a "final" model was believed to have been reached. This implied that the two versions of the model needed to be kept up-to-date continuously, which was a time-consuming and error prone task. Support for multi-lingual process models would have greatly simplified the work. We believe such functionality would also be appreciated by companies operating in multi-lingual environments and utilizing business process technology.

Epilogue

A prototype was developed at Järfälla municipality to prove the applicability of open source BPMSs as back-end systems for e-services. The prototype was

demonstrated to the municipal officials attending the process analysis work. The fact that the solution was not integrated to the present web-portal or database was stressed. It was pointed out that the final implementation will differ from the demonstrated one, as it will be integrated in the municipal current data-centric IT system which offers fundamentally different approach of work support. The officials were genuinely positive to the prototype. They liked the fact that the process really followed the "process map" developed during the workshops. They appreciated that work will be "pushed" to them, which supports the monitoring of the cases they need to carry out today. One of the officials commented:

> "If the new e-applications are handled like this, it will be a great simplification of our work compared to today."

The prototype was also used as input for preparing the requirements specification contracts with the two software development companies in charge of the e-service implementation and the current IT system supplier. In order to accommodate the e-service, the IT system supplier needed to introduce significant changes both into the business IT system they provide and to the municipal database they maintain. The requested changes were too expensive for the municipality to bear which initially made the IT supplier reluctant to take upon the project. After the prototype was developed, the IT supplier decided to change the cost model and invest in the further development of their system. At this point the IT supplier was convinced that this development would not only be beneficial for Järfälla but for all their municipal clients.

The prototype was developed as a proof-of-concept implementation of open source BPMSs as back-end systems for e-services. In practice, however, its major benefit turned out to be the design's applicability to other open e-services. Since applications for other social services are handled similarly to this of the emergency phone application, major parts of the process model were reused for the requirements specification of the two other e-services within the project scope, namely part-time successor and companion services. In addition, the design was found by officials in Järfälla to be *applicable to a major portion of other simple, routine-type cases* in social services such as lunch box and help in the home services (these were outside the scope of our project). The design was also evaluated by the social director and the chief information officer of Järfälla and assessed to be applicable to simple routine-type cases in *other areas of municipal operations* than elderly and disabled care. All in all, these evaluations indicate that the design can be generalized and applied to a large array of open e-services in public administration. This outlines the direction for future work at the municipality.

The prototype shows how the administrative process for simple cases can be supported and automated (within the current Swedish legislation (SoL)). The model is applicable to all Swedish municipalities, as they are all responsible for the provision of the same social services. Sweden is divided in 290 municipalities. There are basically two major municipal IT business systems: one of them is data-centric the other one is process-centric. Järfälla uses the data-centric system and the prototype described here was used for the requirements specification for

the further development of this system. To disseminate the results, the prototype was also shown on a network meeting of 27 municipalities using the competing process-centric IT system. These municipalities were at the time working on a joint requirements specification for the further development of their IT system. The prototype was meant to serve as a source of inspiration for their work. Fortunately and unfortunately, during this meeting the prototype was considered as a "visionary solution". Fortunately, because the strengths of the solution were appreciated, unfortunately because the prototype actually demonstrates the technical possibilities today, not our endeavours tomorrow.

References

1. European Union. Web accessibility policy, http://europa.eu/geninfo/accessibility_policy_en.htm (accessed June 28, 2010)
2. ter Hofstede, A., van der Aalst, W.M.P., Adams, M., Russell, N. (eds.): Modern Business Process Automation: YAWL and its Support Environment. Springer, Heidelberg (2010)
3. Integrations- och jämnställdhetsdepartamentet. Lag (2009:724) om nationella minoriteter och minoritetsspråk (2009) (in Swedish), http://www.notisum.se/rnp/sls/lag/20090724.htm (accessed June 28, 2010)
4. Regeringskansliet, Statens offentliga utredningar (SOU). Strategi för myndigheternas arbete med e-förvaltning (2009) (in Swedish), http://www.regeringen.se/sb/d/11456/a/133813 (accessed July 5, 2010)
5. Statistics Sweden. The future population of Sweden (2009), http://www.scb.se/statistik/_publikationer/BE0401_2009I60_BR_BE51BR0901ENG.pdf (accessed July 15, 2010)
6. Steg, H., Strese, H., Loroff, C., Hull, J., Schmidt, S.: Europe is facing a demographic challenge Ambient Assisted Living offers solutions (2006), http://www.aal169.org/Projectdoc.html (accessed July 22, 2010)
7. Swedish Association of Local Authorities and Regions. Care of the Elderly in Sweden Today 2006 (2007), http://skl.se/web/Publications_and_reports.aspx (accessed July 15, 2010)
8. W3C. Techniques for web content accessibility guidelines 1.0 (2000), http://www.w3.org/TR/WAI-WEBCONTENT-TECHS/#tech-directly-accessible (accessed June 28, 2010)
9. Wohed, P., Russell, N., ter Hofstede, A.H.M., Andersson, B., van der Aalst, W.M.P.: Patterns-based evaluation of open source BPM systems: The cases of jBPM, OpenWFE, and Enhydra Shark. Inf. Softw. Technol. 51(8), 1187–1216 (2009)

In Search for a Good Theory: Commuting between Research and Practice in Business Process Domain

Ilia Bider[1,2]

[1] IbisSoft ab Sweden
ilia@ibissoft.se
[2] DSV, Stockholm University, Sweden

Abstract. Kurt Lewin's statement "There is nothing more practical than a good theory" says not so much about what is good for practice, but rather what it means to have a good theory. There exist a number of competing theories in the business process domain. The current paper is devoted to one of those that lie outside the mainstream direction. The purpose of the paper is not to present the theory as such, but to present the stages of how it was developed with the aim of becoming a "good" theory from the practical point of view. The paper is written as an experience report and goes through different stages of the development where research efforts where intermixed with practical tests. The theory in question is the state-oriented view on business processes. The basic idea of this theory lies in application of the general principles of the theory of dynamic systems to the business domain. The main direction for practical application of theoretical results is the development of IT-support for loosely structured business processes. Besides giving the history of the related research and practical efforts, the paper discusses the lessons learned that can be of interest for the development of other theoretical models/views in the business process domain.

Keywords: business process, theory, practice, dynamic system, state space.

1 Introduction

"There is nothing more practical than a good theory" wrote Kurt Lewin [1], p. 169. Two questions arise in connection to this statement. Firstly, how do we know that the theory is good? Secondly, how to create a "good theory"? As far as as the first question is concerned, we can refer to the story from the Nobel lecture of the famous physicist R. Feynman [2]:

"... One day a dispute arose at a Physical Society meeting as to the correctness of a calculation by Slotnick of the interaction of an electron with a neutron using pseudo scalar theory with pseudo vector coupling and also, pseudo scalar theory with pseudo scalar coupling. He had found that the answers were not the same, in fact, by one theory, the result was divergent, although convergent with the other. ... I went home, and during the evening I worked out the electron neutron scattering for the pseudo scalar and pseudo vector coupling, saw they were not equal and subtracted them, and worked out the difference in detail. The next day at the meeting, I saw Slotnick and said, "Slotnick, I worked it out last night, I wanted to see if I got the same answers

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 16–30, 2011.

you do. I got a different answer for each coupling - but, I would like to check in detail with you because I want to make sure of my methods." And, he said, "what do you mean you worked it out last night, it took me six months!" ... it took him six months to do the case of zero momentum transfer, whereas, during one evening I had done the finite and arbitrary momentum transfer. That was a thrilling moment for me, like receiving the Nobel Prize, because that convinced me, at last, I did have some kind of method and technique and understood how to do something that other people did not know how to do. ..."

The last sentence in the citation above explains exactly what a good theory is. In the light of this explanation the original citation from K. Levin can be interpreted not only as a statement about what is good for practice, but also as a definition of what a good theory is, i.e. a theory that can be useful in practice.

The second question (how to work out a good theory) is not easy to answer, there can be many possible options. Without pretending to give the ultimate answer, the paper presents an experience report on searching for a "good" theory in the business process domain[1] by "commuting" between research and practice. The experience in question concerns the development of the state-oriented approach for controlling business processes. It stretches over a period of more than 25 years, and includes contribution from many people with whom the author cooperated in practice or in research or in both. The search has not been finished yet, but on the way we have worked out and tested in practice a number of hypothesis, some of which still hold and others do not. The author believes that the experience may be useful for others in two respects: (a) as an example of "commuting" between research and practice, (b) which hypothesis have been tested and which of them held and which of them fell (the latter might be more interested than the former)

The paper is structured according to the periods where research or practice where predominant. It starts with the initial theory development (Section 2), goes to the practical test (Section 3) based on which a specialized theory was developed (Section 4) and tested in practice (Section 5).

When the last test in practice showed that we went into the wall, instead of going back to revise the theory, we removed self-imposed theoretical restrictions, and continued experimenting in practice based on the intuition. After the experiments showed that the new direction was promising, we went back to the theory and revised it according to the experience. After the revision, we went back and implemented the results in practice. This latest period of the theory revision is discussed in Section 6. The concluding Section 7 is devoted to general discussion on interconnection between research and practice and lessons learned.

2 Developing the Initial Theory

The event that triggered our long search was as follows. In the beginning of 1980^{th,} the author was responsible for supply of scientific information for the research laboratory he was working for at the time. My duty, in this respect, was to go to the technical library once a week, browse through the new magazines, and make copies of articles

1 Term *domain* is used here to denote: *a sphere of activity, concern, or function; a field.*

that might be of use for the research and practical work conducted inside the laboratory. Sometime in 1984, I came across the survey of Meyrowitz and van Dam [3] on interactive editing. The paper, actually, was more than just a survey as it promoted the idea that an editor should not be a passive program, but an active system helping its users in their work, e.g., by suggesting continuation as in the syntax-driven editors for programming languages.

After reading the survey myself, I gave it to my friend and colleague Maxim Khomyakov, who became so exited that he arranged a serious of discussions on the topic. These discussions, resulted in creating an informal project aimed at developing a theoretical framework for designing of what we then called "human-assisted systems". The idea was explained with the help of two pictures, one representing traditional at the time human-assisting systems (see Fig. 1 to the left), the other one - a new paradigm of "human-assisted systems (see Fig. 1 to the right).

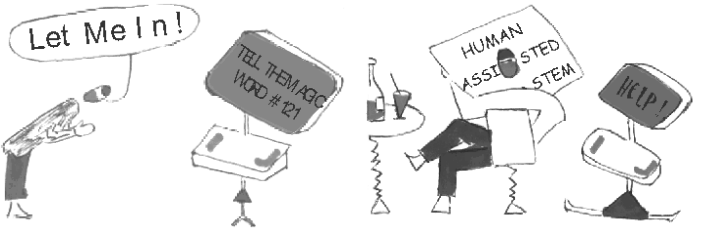


Fig. 1. Human-assisting vs Human-assisted systems

In a *human*-assisting *system* (see Fig.1 to the left), a computer helps a human being to perform certain tasks, e.g. to write a letter, print an invoice, complete a transaction, etc. The relations between these tasks, and the aim of the whole process are beyond the understanding of the computer, but are a prerogative of the human participant. In a *human-assisted system* (see the Fig.1 to the right), the roles are somewhat reversed, the computer has some knowledge about the process and keeps it running as long as it can. When the system cannot perform a task on its own or figure out what to do next, it will ask the human participant for assistance. The human-assisted system frees human beings from tedious, routing work, like searching for information, bookkeeping, reporting, allowing them to concentrate on thing at which they are best, i.e. decision making.

The main idea of human-assisted systems was that the users and the system should work in a symbiosis. The symbiosis should be flexible which means such cooperation between the system and its users where the distribution of responsibilities between them may change in time. It means that the points of interaction between the system and its users may change, thus we need to have a model in which such changes do not require substantial modifications. The latter requires a model in which both human and system actions are represented uniformly on equal footing. This, however, does not imply that the information needed for humans and machines for completing similar operations should be presented to them in the same way.

The project was of theoretical nature. We had one example in mind though, creating a computer programmer's secretary, a system that would help a programmer to managed his/her job, i.e. to ensure that the programmer does not forget to compile and test after making changes in the source code. The "secretary" should considerably extend the capability of the tools like make/build that existed at the time. The project continued for two years from 1984 to 1986 with three main participants, Maxim Khomyakov (initiator), Eugene Pushchinsky, and the author. The result was a model that consisted of the following components:

– a set of atoms,
– a set of objects,
– a code of laws, and
– a set of connectors, each connector hanging on a group of objects that must obey a certain law.

Objects have complex structure expressed by including in their "bodies" a set of connectors that hang on other objects making the latter sub-objects to the former. An object's body can also include a connector hanging on the object itself, as in Fig. 2. The dynamics of the objects-connectors model can be defined by a machine in which a connector is regarded as a processing unit that monitors its operands (objects). A connector

– *awakes* when one of its operands has been changed,
– *checks* whether the law still holds by reading the condition,
– *restores* it when it has been broken,
– *falls asleep*.

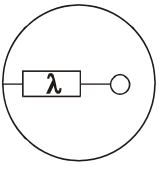


Fig. 2. A connector hanging on an object whose body includes this connector

A law can be fully deterministic, or not. Non-determinism can concern the condition of awakening, or rules of restoring the law, or both. A connector with a non-deterministic law is called a boundary connector. A boundary connector cannot do its job alone and needs help; this is where human beings are introduced in the model. Humans are parts of boundary connectors assigned to help them to decide when to awake, or/and how to restore the state of the objects entrusted to this connector.

As we mentioned above, a connector can both be included in the body of an object, and "hang" on this object (Fig. 2). This allows an object to reconfigure itself based on changes in other objects. Such reconfiguration can include adding new connecters or removing the existing ones, including the one that completes the reconfiguration itself.

The objects-connectors model was published later as a theoretical platform, see [4,5], and [6] (appendixes B and C). The model itself has never been directly

implemented in practice (so far), but served as guidelines for developing specialized theoretical and practical approaches in the domain of business processes.

3 Testing the Initial Theory in Practice

We did not find an occasion to apply the objects-connectors model to a practical task until 1988 when the author started to work for a small Swedish IT-consulting company. The company had developed a system to assist sales staff in the pharmaceutical industry, and my task was to support and farther develop this system.

A salesperson in this industry was driving around the country, meeting doctors at various hospitals and leaving them samples of various (new) medicines. The system was intended to help the salesperson to keep order in his/her business, e.g. plan trips, track samples, e.g., in which hospital they were left, and when it is time to follow up the initial contacts. It also helped to gather statistics and analyze sales potential. The business, hence the system, were built around such concepts as planned activity, execution of activity, planning the follow-up activities, like phone call.

While working with the system, I started to make a model of the sales business in the pharmaceutical domain in terms of objects-connectors framework. In this model, a sales lead on which a sales-person is working is represented by a "sales" object. As sub-objects, it includes a hospital, and a doctor to whom this particular lead is related. A planned activity is represented by a boundary connector included in the body of the sales object and hanging on it at the same time, as in Fig. 2. This boundary connector wakes up at the deadline point and asks the salesperson to complete the activity. While executing the activity, the sales person writes a report and plan new activities. In terms of the model, the boundary connector that represents the planned activity removes itself from the body of the object placing at the same time some other connectors (new planned activities) instead of it. Part of this activities could be calculated based on the rules, others are to be added manually by the salesperson behind the "steering wheel" of the boundary connector.

This model served later as basis for a new project that we started in 1989 after founding a new consulting company IbisSoft. The project was called DealDriver and was aimed a creating a system to support sales and order processing. The DealDriver was initially meant for a company selling goods via telephone. The system was to support selling via phone calls, following up customers, and taking and processing orders via phone. The latter included packing, delivery, invoicing, reminding, and payment registration. The selling process was a lighter version of the above description. For order processing we design a special object that represented the current state of processing. This object was presented to the end-users as a screen in Fig. 3.

The screen in Fig. 3 presents to the user how much of the order has been processed so far, what has been ordered, how much of ordered goods have been delivered, whether some money has been invoiced and/or payed. This screen represents a so-called static part of the order object. This static part is complemented by a dynamic part represented in Fig. 4. To the end-user this part is shown as a plan of activities that corresponds to the state of the order processing. In our model, a plan conceptually represent a set of boundary connectors fired by their deadlines to complete some work with the human assistance. This work includes removing the boundary connector

itself from the plan, and adding some new boundary connectors instead of it based on the emerging state of the static part of the order object. These new connectors/activities could be planned automatically, or added manually by the end-user. DealDriver included hard-corded automatic planning of next steps, e.g., from order, to packing and delivery from delivery to invoicing (or rest delivery), from invoicing to registering payment (or sending a reminder), etc. In addition DealDriver allowed its uses to add new activities manually.

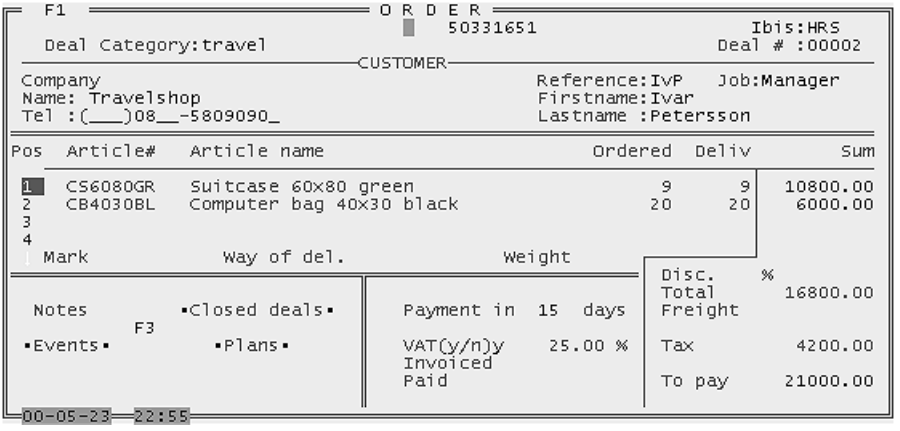


Fig. 3. Screen representing the static part of the order object

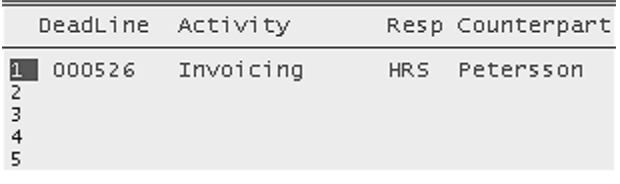


Fig. 4. Screen representing the dynamic part of the order object

Though DealDriver itself was not especially successful in the market, it has been used internally at IbisSoft for more than ten years. Besides, several derivatives of DealDriver developed by the team of my colleague Maxim Khomyakov enjoyed some level of success in Russia. One of these derivatives even won the “Object Applications of the Year Awards 1997” in the group "Best object-based application developed using non object-oriented tools" (Object World Show in London, April 1997). More on our experience of this period see in [7,8,9].

4 Developing the State-Oriented View on Business Processes

Based on our experience of the DealDriver project and its derivatives, we developed a state-oriented approach to business process modeling and control. This approach does

not directly refer to our initial objects-connectors model that influenced it. Besides the objects-connectors framework, the method was influenced by the ideas from the Mathematical systems theory devoted to modeling and controlling dynamical systems in the physical world [10]. The main concept of the state-oriented view on business processes is a *state* of the process instance that can be defined as a position in some state space. A state space is considered multidimensional, where each dimension represents some important parameter (and its possible values) of the business process. Each point in the state space represents a possible result of the execution of a process instance. If we add the time axis to the state space, then a trajectory (curve) in the space-time will represent a possible execution of a process instance in time. A process type is defined as a subset of allowed trajectories in space-time.

Consider, for example, order processing as in Fig. 3; the numeric dimensions that can be used for the state space for this process can be defined as follows:

- In the first place there are a number of pairs of product-related dimensions <ordered, delivered>, one pair for each product being sold. The first dimension represents the number of ordered items of a particular product. The second one represents the number of already delivered items of this product. The number of such pairs of dimensions is not fixed but is less than or equal to the size of the company's product assortment.
- In addition, there are two numeric dimensions concerning payment: invoiced (amount of money invoiced) and paid (amount of money already received from the customer).

Each process instance of the given type has a goal that can be defined as a set of conditions that have to be fulfilled before a process instance can be considered as finished (end of the process instance trajectory in the space state). A state that satisfies these conditions is called a *final state* of the process. The set of final states for the process in Fig. 3 can be defined as follows: (a) for each ordered item *Ordered* = *Delivered*; (b) *To pay* = *Total* + *Freight* + *Tax*; (c) *Invoiced* = *To pay*; (d) *Paid* = *Invoiced*. These conditions define a "surface" in the state space of this process type.

The process instance is driven forward through *activities* executed either automatically or with a human assistance. Activities can be planned first and executed later. A *planned activity* records such information as type of action (goods shipment, compiling a program, sending a letter), planned date and time, deadline, name of a person responsible for an action, etc.

All activities planned and executed in the frame of the process should be aimed at diminishing the *distance* between the current position in the state space and the *nearest* final state. The meaning of the term distance depends on the business process in question. Here, we use this term informally. For example, activities to plan for the process in Fig. 3 can be defined in the following manner:

- If for some item *Ordered* > *Delivered*, *shipment* should be performed, or
- If *To pay* > *Invoiced*, an *invoice* should be sent, etc.

All activities currently planned for a process instance make up the process *plan*. The plan together with the current position in the state space constitute a so-called generalized state of the process, the plan being an "active" part of it (engine). The process plan on Fig. 4 corresponds to the process instance state shown in Fig. 3. The plan

plays the same role as the derivatives in the formalisms used for modeling and controlling physical processes in the Mathematical systems theory. Planned activities shows the direction (type of action) and speed of movement (deadlines), i.e. exactly what the first derivatives do in the continues state space.

With regards to the generalized state, the notion of a *valid* state can be defined in addition to the notion of *final state*. To be valid, the generalized state should include all activities required for moving the process to the next state towards the goal. A business process type can be defined as a set of valid generalized states. A business process instance is considered as belonging to this type if for any given moment of time its generalized state belongs to this set. This definition can be converted into an operational procedure called *rules of planning*. The rules specify what activities could/should be added to/deleted from an invalid generalized state to make it valid. Using these rules, the process instance is driven forward in the following manner. First, an activity from the plan is executed and the position of the process instance in the state space is changed. Then, the plan is corrected to make the generalized state valid; as a result, new activities may be added to the plan, and some existing activities can be modified, or removed from the plan.

The idea of the state oriented view on business processes was first presented in 2000 [11]. For more details on this view see [12],[13],[14].

5 Back to Practice

The state-oriented view on business processes discussed in the previous section was tested in practice for two tasks:

– Business process modeling
– Development of business process support systems

As far as business process modeling is concerned, we developed a method of process modeling alternative to drawing activities/workflow diagrams. The main steps in this methods are as follows:

– Designing a state space for a given business process using mockup screens of the kind of Fig. 3,4
– Listing activities to be completed in the process. Each activity has informal definition that includes: in what state of the process it should/could be planned or executed, what should be done during its execution (what state will emerge after the execution), who should be completing the activity (roles).
– Creating examples of the process instances run. For this end we designed a special tool called Process Visualizer. Using this tool, the user could create a trace of a process instance trajectory by inputting values in mockup screens. Besides the state screen, we used a plan screen (as in Fig. 4), and an event screen (completed activities). After building such a trajectory, one could follow it step by step in forward or backwards direction seeing how planned activities are converted into the state changes and new activities planned.

We have completed over 10 of such process modeling projects in the public sector and non-for-profit organizations. We worked mostly with modeling of so called loosely structured processes[2], like decision-making, recruiting, negotiations, investigations, etc. Results from some of these project were published in [15,16]. Our experience showed that the state-oriented approach was quite suitable for modeling loosely structured process and the resulting models could be easily understood and evaluated by business experts who participated in these projects. Details on why the state-oriented approach is particularly suitable for modeling loosely structured processes see in [17], and partly [16].

For most of the modeling projects (but not all of them) the goal was twofold: (a) to identify and understand the processes, and (b) create requirements for a business process support system. For some of the processes we analyzed these requirements were implemented in our new system called ProBis developed for a Swedish interest organization in 2003-2006 as described in [18]. ProBis continued the architectural ideas of DealDriver underpinned by the state-oriented view on business processes and a rich graphical environment (DealDriver was developed for character-based displays). Much more attention was payed for creating a convenient environment for manual collaborative planning. ProBis included support for a number of different processes, but used a standardized way of presenting the generalized state of the current instance to the end-users. Below we give a short overview of ProBis, for more details see in [1,19,13,14]..

The generalized state of the process in ProBis is presented to the end-user as a window divided in several areas by using the tab dialogues technique, see Fig. 5. Some areas of the window are standard, i.e. independent from the type of the business process; 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 the generalized state presentation in ProBis includes also the task area (tab) that contains two lists, as in Fig. 5. The *to-do* list (to the left on Fig. 5) includes tasks (synonym for activities from Sections 3,4) planned for the given process instance; the *done* list (to the right on Fig. 5) 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. All tasks planned for a given person from all process instances are shown in the end-user's personal calendar. From the calendar, the user can go to any process instance for which he has a task assigned to him/her in order to inspect, change, or execute this task.

Tasks in iPB are used as a way of communication between process participants. The communication is performed by assigning a task to another user via filling a special task form. One chooses the task from the list, assigns it to another ProBis user, adds a textual description, and some parameters, for example, attaches a document

[2] Under loosely structured we mean a process for which the order of activities/tasks is difficult or impossible to establish.

that is already placed in the process instance space. The task list is configurable and can be adjusted for each installation and process type.

To further facilitate communication, several more advanced features were added to ProBis. For example, there is a possibility to plan the same task to many users. Another advanced feature is the "Returned receipt" check-box which ensures that the planner gets a special "Attention" task planned for him as soon as the task he/she has assigned to somebody else has been completed.

Our experience with ProBis shows that this kind of systems is quite suitable for loosely structured processes when used by a professional team that knows how to use the system quite well. Introduction of such system into operational practice especially with many occasional users is a challenge (see [18,19]).

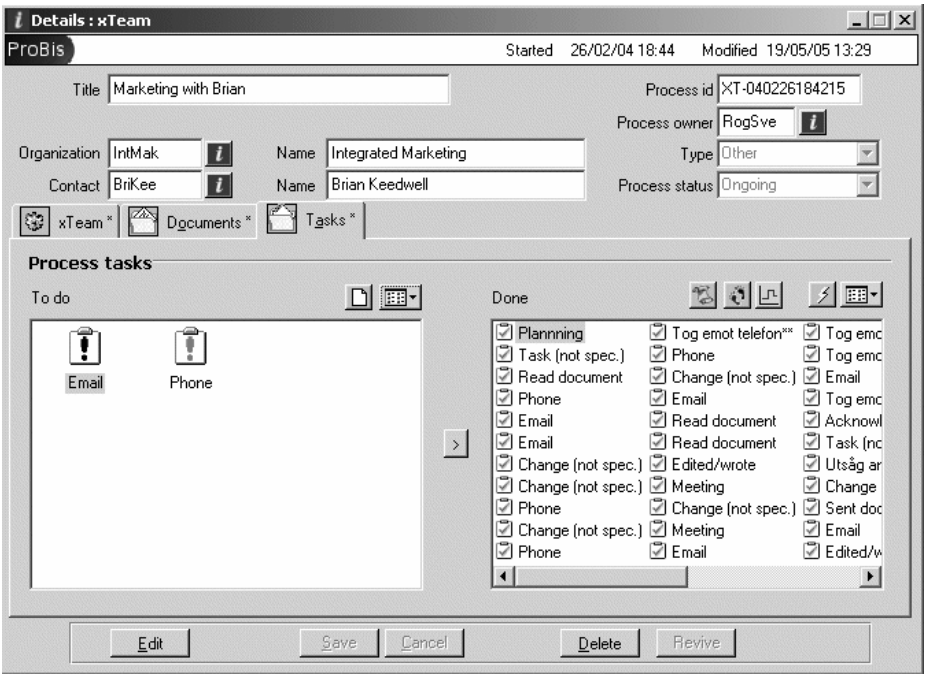


Fig. 5. View on the generalized state in ProBis

We found two main drawbacks with ProBis when using it for more structured process or/and processes that involve occasional users:

- 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.
- 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 communication 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.

6 New Revision

A new period in our practical and theoretical development started when we moved from the Windows environment to WEB in connection to the development of a new business process support system for the municipality of Jonköping. The system was aimed at supporting investigations on suspected child abuse. Instead of porting ProBis to a new technical platform, we started from scratch based on our practical experience. Several decision were made in the beginning of the new project:

- Instead of developing a process support system we decided to create a tool that allows the non-technical people to define their own processes and automatically get web-based support for them.
- We dropped the idea of using low-level activities (tasks) planning as a primary mechanism for driving the process forward.
- We abandon the idea of using tab dialogs for structuring the state space. Instead, we accepted the idea of grouping state parameter (dimensions) in blocks based on the order in which their value are to be obtained.

After the initial test in practice, we discovered that the end-users liked the system and considered the sequence of blocks of parameters as a kind of process map. Based on the positive feedback, we revised our theoretical ideas and implemented the revised version in the tool. This work has not been finished yet, and we continue to revise both theoretical concepts and their practical implementation.

The tool got the name iPB (interactive Process applications Builder) [19,20,21]. Its purpose is to serve as a platform for building support for relatively loosely structured business processes. The goal is reached via creating a process definition for each process that requires computerized support. The definition is built around the process map; the map is defined as a drawing that consists of boxes placed in some order. Each box represents a step of the process, and the name of the step appears 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.

Each process instance/case gets its own copy of the map, see Fig. 6. The map is used for multiple purposes: as an overview of the instance/case, guidelines for handling it, and a menu for navigating inside the state space. The user navigates through the state space by clicking on the boxes of the steps. 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. These constraints are defined with the help of so-called business rules. A click on a step box redirects the end-user to a web-form that assists him/her in completing the step. The form contains text fields, option menus and radio-buttons to make choices, check boxes, as well as more complex fields. The form may also include “static” explanatory texts which instruct the users how to fill the form , and thus reduce the needs for lengthy manuals.

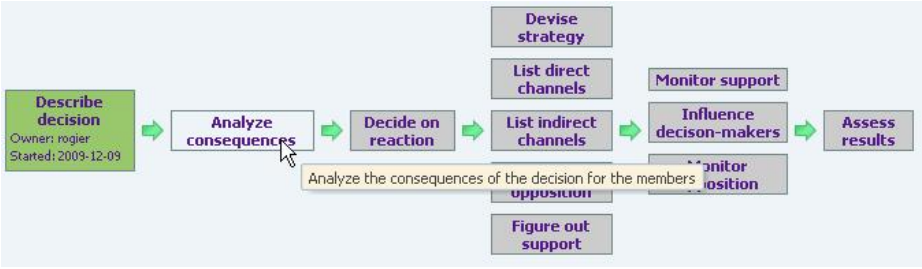


Fig. 6. A map used for structuring the state space in iPB

The progress in filling the step forms is reflected in the map 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.

From the theoretical point of view, the approach described above represents a modification of our state-oriented view on business processes. The basic ideas behind this modification consist in 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. 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 (with artificial arrows showing the left to right direction (Fig. 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.

7 Discussion and Lessons Learned

In the previous sections, we described a journey aimed at developing a “good” theory in the business process domain in the sense that was define in Introduction. The journey has not been finished yet, and whether it has been successful or not depends on the subjective judgment. However, this is an experience that can be analyzed even at this stage and both positive and negative lessons can be drawn from it.

During all stages of the journey, we had some practical aims/examples on which various hypothesis where tested. Even on the initial theoretical stage, we had a "programmer's secretary" in mind when debating different alternatives in building a theoretical framework. Without this condition, the journey discussed in this paper could not have achieved any progress.

Having a highly abstract theoretical framework in the back of our minds gave us a possibility to see things in reality differently from what they were on the surface. Without having an objects-connectors model from Section 2, it would be impossible for us to analyze the sales business activity in the way that lead to the development of the state-oriented view on business processes. Such high-level abstract models as the objects-connectors one is difficult to apply directly as they are, but they serve as perfect guidelines to see the world in an unusual perspective.

Our state-oriented view on business processes was influenced by the Mathematical systems theory that investigates processes in the physical world. Transferring the ideas from one domain to another quite often lead to good theories and is recommended by the general systems thinking [22]. However direct "borrowing" a theory from one domain into another will hardly work. In our case we borrowed only the ideas of movement in a state space, without adopting the apparatus of differential equation or state-transition diagrams. "Borrowing" should go on the conceptual level, the details should be designed taking into consideration the nature of the new domain.

A straightforward application of the theory to practice might not always work in the business process domain. While using planning for driving the process forward was quite natural for people working in the sales department, it was considered as counter-intuitive for the processes participants in other domains. While making people to accept a completely new way of seeing the world is possible, this creates a considerable hinder for introducing new ideas and techniques that is best to avoid.

Our experience with iPB shows that its way to present and support business processes is considered quite natural by many people. In our opinion, two iPB properties contribute to this. Firstly, it uses highly visualized and very simple process map to represents the overall position in the state space to all people participating in the process. Secondly, it uses´the idea of forms to represent the details. A form is a familiar concept for most modern societies and thus can be easily understood by, practically, everybody. The lesson learned here is that applying a theory in practice where people, not machines, are the main driving force requires presenting new ideas in a familiar to them form. The latter gives better chances for success of the application of a new theory in practice.

At the end of the ProBis period, we found that introducing this kind of systems is (even when possible) creates certain difficulties on the floor. Instead of trying to find a theoretical solution, we loosen our attachment to our theory and started free experimenting in practice. After this experimenting gave some positive results, we went back to adjust the theory. Two lessons can be drawn from this experience. Firstly, narrow following the theory may lead to the dead-en. Secondly, free experiment can give an idea how to develop the theory.

Our experience has also shown that a proper theory will stands, even when the facts are against it. To make a theory compatible with the facts, one may need to abandon some of its narrow "axioms" [22]. In our case, it was the idea that using what we call (detailed) dynamic distributed planning is mandatory for applying our state-oriented view on business processes to practice. Freeing ourselves from this "axiom" and concentrating on the interplay of subspaces, in a way, "saved" the theory.

Our theoretical and practical thinking was influenced by a number of ideas and systems that we found in other domains, e.g. mathematical system theory, programming language REFAL, the general systems thinking to name a few. None of the ideas that has influenced our work, however, has been taken from the business process domain. Not following the mainstream in the domain, helped us to be outside the workflow box and see alternative solutions for what we call loosely structured business processes, for which the mainstream does not have any good solutions. The lesson learned here is that following too much what is happening in one's own domain may be harmful for creativity. Looking beyond it into the adjustment domains may give a better clue how to continue in one's own domain.

The above does not mean that we did not know, or rejected everything that had been done in the business process domain. In [17] we classified different views on business processes, and pointed out the areas of applicability for each of them. We believe that each view has its own area of application and should be developed and tested according to it, and we hope that the lessons listed above could be useful even when developing these views.

In conclusion we want to mention a more general principle that we have followed on our journey. There are two ways of scientific investigation of reality:

– With minimum interference, i.e. through observation and measurement
– With maximum interference, i.e. by applying a considerable force and seeing how the object under investigation behaves.

While both methods are legitimate and widely used, there is a practical limit of what can be achieved through using only the first one. If we deal with a complex system (like an enterprise, or a governmental office), its structure and behavior may not reveal itself unless under a stress. This is why we have chosen the second method combining research and practice in the setting of a consulting company whose customers were from time to time interested and willing to try some new ideas in their organizational practice. The latter always means an organizational change that, usually, puts the system (organization) under stress.

It is difficult to follow the second method (with maximum experience) why working inside the university walls. The research groups in business process domain should find their way to get into practical surrounding to conduct proper research according to the method of maximum interference.

Acknowledgments. This paper would have never been written without considerable efforts of the team of researchers, developers, business-experts and end-users with whom the author worked in research and practice. Not having the space to name all of the participants of the long journey, I especially wish to express my gratitude to my nearest colleagues Tomas Andersson, Alexander Durnovo, Paul Johannesson, Maxim Khomyakov, Erik Perjons, Eugene Pushchinsky, Alexey Striy, Rogier Svensson.

References

[1] Lewin, K.: Field theory in social science: Selected theoretical papers by Kurt Lewin. Tavistock, London (1952)
[2] Feyman, R.P.: Nobel Lecture, December 11 (1965), http://nobelprize.org/nobel_prizes/physics/laureates/1965/feynman-lecture.html

[3] Meyrowitz, N., van Dam, A.: Interactive Editing Systems. ACM Computing Surveys 14(3), 321–415 (1982)
[4] Bider, I., Khomyakov, M., Pushchinsky, E.: Logic of change: Semantics of object systems with active relations. Automated Software Engineering (ASE) 7(1), 9–37 (2000)
[5] Bider, I., Khomyakov, M.: New technology - Great Opportunities. How to Exploit Them. In: Filipe, J. (ed.) Enterprise Information Systems IV, pp. 11–20. Kluwer, Dordrecht (2003)
[6] Bider, I.: State-oriented business process modeling: principles, theory and practice. PhD thesis, KTH (Royal Institute of Technology), Stockholm (2002)
[7] Bider, I.: Developing tool support for process oriented management. In: Handbook of Systems Development, vol. 1999, pp. 205–222. CRC Press, Boca Raton (1998)
[8] Bider, I.: Object driver: a method for analysis, design, and implementation of interactive applications. In: Handbook of Systems Development, vol. 1999, pp. 81–96. CRC Press, Boca Raton (1998)
[9] Bider, I., Khomyakov, M.: Object-oriented model for representing software production processes. In: Dannenberg, R.B., Mitchell, S. (eds.) ECOOP 1997 Workshops. LNCS, vol. 1357, pp. 319–322. Springer, Heidelberg (1998)
[10] Kalman, R.E., Falb, P.L., Arbib, M.A.: Topics in Mathematical System Theory. McGraw-Hill, New York (1969)
[11] Khomyakov, M., Bider, I.: Achieving Workflow Flexibility through Taming the Chaos. In: OOIS 2000-6th International Conference on Object Oriented Information Systems, pp. 85–92. Springer, Heidelberg (2000)
[12] Andersson, B., Bider, I., Johannesson, P., Perjons, E.: Towards a Formal Definition of Goal-Oriented Business Process Patterns. BPMJ 11(6), 650–662 (2005)
[13] Regev, G., Bider, I., Wegmann, A.: Defining Business Process Flexibility with the help of Invariants. SPIP 12(1), 65–79 (2007)
[14] Bider, I., Striy, A.: Controlling business process instance flexibility via rules of planning. IJBPIM 3(1), 15–25 (2008)
[15] Andersson, T., Andersson-Ceder, A., Bider, I.: State Flow as a Way of Analyzing Business Processes – Case Studies. Logistics Information Management 15(1), 34–45 (2002)
[16] Perjons, E., Bider, I., Andersson, B.: Building and Exploiting a Business Process Model for Lobbying: Experience Report. Communications of the IIMA (CIIMA) 7(3), 1–14 (2007)
[17] Bider, I., Perjons, E.: Evaluating Adequacy of Business Process Modeling Approaches. In: Handbook of Research on Complex Dynamic Process Management: Techniques for Adaptability in Turbulent Environments, pp. 79–102. IGI (2009)
[18] Andersson, T., Bider, I., Svensson, R.: Aligning people to business processes experience report. Software Process Improvement and Practice (SPIP) 10(4), 403–413 (2005)
[19] Bider, I., Perjons, E., Johannesson, P.: In Search of the Holy Grail: Integrating social software with BPM. In: Experience Report, Enterprise, Business-Process and Information Systems Modeling. LNBIP, vol. 50, pp. 1–13. Springer, Heidelberg (2010)
[20] Bider, I., Perjons, E., Johannesson, P.: A strategy for integrating social software with business process support. LNBIP, vol. 66, p. 12. Springer, Heidelberg (2011)
[21] iPB Reference Manual on-line documentation, `http://docs.ibissoft.se/node/3`. [Online] (accessed June 20, 2010)
[22] Weinberg, G.M.: An Introduction to General Systems Thinking. Dorset House, New York (2001)

Does Process Mining Add to Internal Auditing? An Experience Report

Mieke Jans*, Benoît Depaire, and Koen Vanhoof

Faculty of Business Economics
Hasselt University
3590 Diepenbeek – Belgium
{mieke.jans,benoit.depaire,koen.vanhoof}@uhasselt.be

Abstract. In this paper we report on our experiences of applying business process mining in a real business context. The context for the application is using process mining for the purpose of internal auditing of a procurement cycle in a large multinational financial institution. One of the targeted outcomes of an internal audit is often the reporting on internal controls over financial reporting (ICFR), since this reporting is mandatory for Sarbanes-Oxley regulated organisations. Our process mining analyses resulted in more identified issues concerning ICFR than the traditional auditing approach. Issues that were identified using process mining analysis concerned violations of the segregation of duties principle, payments without approval, and violations of company specific internal procedures.

Keywords: process mining, internal audit, control monitoring.

1 Introduction

In this report we present our experiences on applying process mining in the context of internal audit. Business process mining, or in short process mining, is a relatively new research domain that aims at discovering a process model based on real transaction logs in the system. There are three main characteristics of process mining that make process mining a viable support to the internal audit. The first characteristic is the use of log data beyond the auditee's influence, what we call meta-data. This enables the auditor to objectively reconstruct the executed process that precedes a transaction (for instance paying an invoice). The second characteristic is the process view it utilizes. The auditor does not need to rely anymore on the designed process model, but can use the discovered, actual process model, as a start for his further procedures. The third valuable characteristic of process mining in an internal audit context is the ability to broaden the scope to tests of both controls and details. Up till now, the use of data analysis in the context of internal auditing, was limited to monitor controls.

* Corresponding author.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 31–45, 2011.

However, when these internal controls are relaxed to allow companies to operate in a more flexible and efficient manner, only monitoring the internal controls provides no assurance on the whole process execution. Hence a process view approach is required to analyze deviations from the designed model, which is made possible through process mining.

We report on a project, conducted at a financial services provider in Europe, where the procurement process is audited using process mining techniques. The analyses were grouped in four types of analyses. In a first group we assemble analyses that focus on the sequence of activities in the process. In the second group we analyze the roles of persons, like for example the roles of an approver or a creator of a purchase order. In a third group of analyses we run some more detailed tests in order to verify certain assertions. This is called the verification phase. In the fourth group of tests, some social networks were examined. A social network within an organization is considered to exist between persons working together on a common case. Prior to our monitoring activities, a traditional internal audit has taken place.

The results of the process mining analyses led to the identification of internal control issues that were not identified during the traditional internal control. The internal control issues concerned payments without approval, violations of the segregation of duty principle, and violations of company specific internal procedures. The identification of additional internal control issues is in line with the findings of Masli et al. [1] where the implementation of internal control monitoring technology is associated with a lower likelihood of material weaknesses.

The remainder of the report is organized as follows. In the next section, some background is provided on process mining, a business process focus in auditing, and continuous auditing. The third section presents the methodology of the study. The fourth and fifth sections report on two preparatory steps: the procurement process analysis and the event log creation. The sixth section reports on the process mining results and we conclude in the last section.

2 Background

In this paper we examine the value that process mining can provide to internal auditors. The Business Process Management Center[1] describes process mining in the following terms:

> "The basic idea of process mining is to extract knowledge from event logs recorded by an information system. Until recently, the information in these event logs was rarely used to analyze the underlying processes. Process mining aims at improving this by providing techniques and tools

[1] The BPM Center is a collaboration between the Information Systems groups (IS@CS and IS@IEIS) at Eindhoven University of Technology (Eindhoven, Netherlands) and the BPM group at the Faculty of Information Technology of Queensland University of Technology (Brisbane, Australia).

> for discovering process, control, data, organizational, and social structures from event logs. Fueled by the omnipresence of event logs in transactional information systems [...] process mining has become a vivid research area." [2]

As the quote indicates, the source of data for process mining is an event log, often referred to as "history", "audit trail", "transaction log", etc. [3] Most businesses of any significant size today store their data, including log data, electronically thanks to the maturing of technologies for databases and computer networks. Process mining is a term subsuming all methods of distilling structured process descriptions from a set of real executions, using log data. [4]

The event log that is used in process mining is to be extracted from the information system (IS). There are conditions that the data, captured in the IS, must meet in order to mine the process. More precisely there are four characteristics that need to be extracted from the IS about each event. An event 1) refers to an *activity*, 2) refers to a *case* - or *process instance*-, 3) can be appointed to a person, the *originator*, and 4) is performed on a certain moment in time, the *timestamp*. For example, the event with unique identifier 000001 refers to a *sign* (activity) of *purchase order 4603* (case) by *Ann Smith* (originator) on *February 5th 2011* (timestamp). These four characteristics are logged by the IS (in most cases the enterprise resource planning (ERP) system) and are beyond the influence of the employee. In the context of this article, this data is referred to as meta-data as it records information, i.e. *activity*, *case*, *originator* and *timestamp*, about the registration of the real data, i.e. the purchase order.

3 Methodology

In order to examine the added value of process mining for internal auditing, an internal audit of the procurement process at a large multinational European bank is conducted, hereby using process mining techniques. The authors are external researchers who were in the position to have access to the relevant data as well as to the internal auditors. The process of procurement is selected for examination as it is a typical, standardized process in most organizations, claiming a large part of the income statement costs (even in a non-trading company[2]), and involving financial reporting activities. The selection of the process is also influenced by the availability of log data. The procurement department is subjected to a standard internal audit right before we applied process mining. This way we can compare the uncovered issues.

Before mining a process, the designed process needs to be clear for the auditor in order to identify which activities constitute the process. To this end, a process analysis is executed. To collect the desired information, various methods are applied. Executive officers (both business and Information Systems experts) were interviewed, employees at various departments were questioned and observed

[2] The case company purchases annually for around 1.4 billion euros.

during their job, and internal user guidelines of the ERP system were consulted. The outcome of the process analysis is presented in the next section.

After the process analysis is executed, an appropriate event log is built from the stored log data. The relevant log data that is captured by the SAP ERP system is vast in magnitude and dispersed over numerous tables (with a certain logic schema depending on the ERP system and company settings). In order to mine this data, the data needs to be configured into an event log with format requirements that are inherent in process mining. This format structures the relevant data around the activities that constitute the process. In our study, all invoices of January 2007 are traced back to their accompanying purchase orders. These purchase orders are then the cases we follow throughout the procurement process. We did not take a sample of this data, but analyzed the whole population. This approach provides prove of the applicability of these techniques to run on a population set on a monthly basis. We discuss the creation of the event log more in detail in section 5.

Once the event log is created, the process is audited by applying several process mining techniques. We divide the analyses into four categories: analyzing the sequence of activities, resulting in "patterns"; a role analysis: analyzing how activities are linked to persons; a verification phase: analyses to further examine the output of the previous steps; and social network analyses.

4 Procurement Process Analysis

In the case company, the procurement process is embedded in the ERP system and is structured as depicted in the flow chart diagram in Figure 1. The process is triggered by the creation of a purchase order (PO hereafter) by a person at the Purchasing Cell. This PO gets signed and released by two distinct persons, herewith approving the order. Once the release has taken place, the employee at the purchasing cell can order the goods with the supplier. This takes place outside the ERP system and is accordingly not depicted in the flow chart. The supplier sends the goods and the accompanying invoice. If both the documents 'Goods Receipt' and 'Invoice' are entered into the system, the accounting department will book the invoice. This last activity will trigger a payment. The payment occurs without any human influence and is stored in another information system and is for this reason not seen as a different activity from the 'book invoice' activity. We therefore depicted activity 6 as "Pay" instead of "book invoice". The system further allows changes in the PO, sometimes triggering a new approval. This is however not depicted in Figure 1, since this does not constitute the optimal, designed process.

The depicted process in Figure 1 is the designed process model. When executing this process in practice, the process can deviate from this prescribed model. For example, some changes can be made to the PO after its creation, perhaps triggering a new 'sign' and 'release' activity. This would result in an extra activity that is not in the process model ('change PO') plus an extra arrow that redirects to the activities 'sign' and 'release'. Another example is the receipt of goods in multiple deliveries. This would cause extra activities of 'receive

goods', perhaps followed by multiple invoices, hence extra activities of 'receive invoice'. One could suggest to configure the ERP settings in such a way that these deviations are not supported nor executable by the system, but this would lead to inoperability and inflexibility which is often not acceptable. Therefore, real process executions can deviate strongly from the designed process and only monitoring the internal controls (that allow for these deviations for the purpose of operational reasons) does not reveal deviations from the designed model. Note that exceptions from the designed model are not per definition internal control failures. Some deviating process executions are normal, other are suboptimal, other are outliers. Tests of details are required to analyze this. In this case study, we execute these tests on the whole population, including the application of a process view.

5 Event Log Creation

In a first stage of building the event log, two preparatory questions need to be resolved. 1) what are the activities that constitute the process? and 2) what is the case or process instance you will follow throughout the process by linking these cases to the key activities?

The activities we select to include in the event log in this case study are based on the information gathered during the process analysis step. The six activities, represented by the six rectangles in Figure 1 are selected to incorporate in the event log. We identify one additional activity: to execute a change. This activity

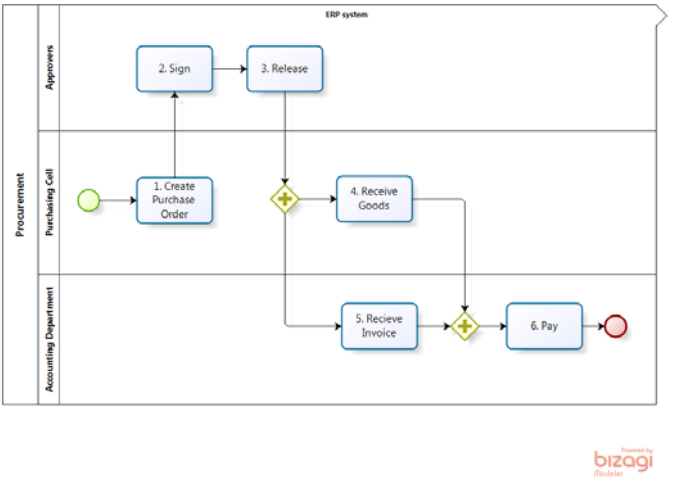


Fig. 1. Flow chart of procurement process at the case company

is not represented in the designed process model, but is available for execution in the ERP system. We incorporate this activity also in the event log. Relevant data on the seven activities (timestamp and originator) was located in SAP, assuring the necessary data availability before we continue.

Aside from the selection of activities, a process instance needs to be selected. A process instance is a case or subject that can be uniquely identified and followed throughout the process by linking this unique id to the subsequent activities it undergoes. For our process instance choice, we selected a line item of a PO to follow throughout the process. We prefer this detailed level of a PO item line over a PO as a whole, because the booking of an invoice in the financial ledger is based on the item line level. However, the activities 'create PO', 'sign', and 'release' refer to the parent PO that the item line belongs to, since there is no timestamp available on the creation of an item line, and since signs and releases are executed on a PO level, as said before. This is not an optimal situation, but given the double dimensionality of the PO -the PO as a whole and the PO as a combination of item lines- that is prosecuted in SAP, we are limited to this solution. The auditor has to bear these limitations in mind when analyzing the results.

For each activity in the abovementioned selection, the timestamp and originator are extracted from the ERP system and a link is made to the according process instance (PO item line). This way an activity flow, called an audit trail, is stored for each process instance under investigation. An audit trail of a case can look as follows: *Create PO - Sign - Release - GR - Change Line - IR - Pay.* Aside from information on the timestamp and originator, extra attributes are stored. The attributes are to be divided in two groups. There are attributes of the process instance (f.i. the value of the PO item line, the purchasing group it belongs to, etc.) and attributes of the activities. The latter attributes depend on the activity it relates to. For instance an attribute of the activity 'Change Line' is the modification that took place (in case of modifying the PO value, otherwise zero), while of the activity 'Goods Receipt' the value and the quantity of the GR, and a reference number to the accompanying invoice are stored as attributes.

Our data download from SAP contained all invoices of January 2007. The invoices were all of the type 'with PO', since this type of invoice and its related process was the process under investigation. The invoices were then traced back to their accompanying PO's. (We found no invoices without PO in this stage.) These PO's, with creation dates between 2005 and 2007, were then followed from their start activity 'Create PO' to their end activity. If the end activity was not a payment, the ending activities were cut off to the last payment activity. This way, we only retained completed procurement cycles. This allows us to identify anomalies in completed cycles. Otherwise the open orders are classified as anomaly, which will not be the intention of an auditor.

6 Process Mining Results

6.1 Log Summary

The scope of our event log are all the PO's that resulted in an invoice in January 2007. This led to 26 185 process instances, containing 181 845 activities, involving 272 originators. At minimum, an audit trail consisted out of four events, on average the audit trail entailed six events, and at maximum 390 events. This maximum amount of events is an outlier that would draw immediate attention to an auditor. It appears to concern an open order, which is used all over again. This does not mean there are per definition anomalies in this case. The auditor may want to analyze this long audit trail more closely by an in depth investigation. The occurrences of the events are summarized in Table 1. In theory, all activities should occur the same number of times. In practice however, the percentages of relative occurrences differ. The number of cases in our event log (26 185) equals the activity of 'Create PO', since this activity refers to the creation of the parent PO that this PO item line belongs to. This activity occurs exactly once per case. Table 1 shows that there are less sign activities than PO (lines) created, meaning that not every PO is signed. On the other hand are there more releases than PO lines. We also identify more payments than cases, implying multiple payments on one case.

Table 1. Occurrences of the events in the event log

Activity	Occurrences (absolute)	Occurrences (relative)
1. Create PO	26 185	14.4%
2. Sign	25 648	14.104%
3. Release	28 748	15.809%
4. GR	24 724	13.596%
5. IR	29 255	16.088%
6. Pay	31 817	17.497%
Change Line	15 468	8.506%

6.2 Tests of Controls and Tests of Details

We categorize our testing into four groups. In the first category, we report all analyses that focus on activity sequences, called activity patterns. In the second category originators are linked to the activities they execute, hereby referring to their role. In this category, analyses concerning segregation of duties are grouped together. In the third category, we group all tests that examine specific attributes of cases or activities, and not merely the activity flow or originator role. The input for these analyses are the results of the previous tests, requiring some verification. Therefore we call this third group of analyses the verification phase. In the last category, the linkages between originators are analyzed, resulting in social networks of the employees involved.

Activity Patterns. Before starting with a process discovery algorithm that extracts a process model from our event log, we report on the patterns and their frequencies as they occur in the event log. The Performance Sequence

Analysis reveals 304 patterns, an extremely high number of patterns for a rather straightforward designed process.

This type of information is unique to process mining, since it utilizes the meta-data on activities and timestamps for calculation. Using traditional analysis techniques would not yield this information. The auditor is probably interested in this high number of patterns for a relatively simple designed process. This way he is cautioned that the process is far more complex than expected. The six most frequent patterns are displayed in Table 2, but of course an auditor may be willing to look at the other patterns too.

Notwithstanding the high number of patterns in this event log, we find a small number of patterns, three out of 304, to represent 80% of the data set. At the other end of the spectrum there are 104 patterns (patterns 200 to 303) that only occur once.

Linking the six patterns to the designed model in Figure 1, we recognize in pattern 1 the model as it is designed. Pattern 2 differs from this pattern in the additional activity that the case (item line) is changed. As discussed before, this is not an outlier or anomaly, but an excepted process execution. In patterns 3 and 4 however, the sign activity is discarded. This is not part of the designed model, but when discussed with the domain expert it appears that there are conditions which can be met in order to allow the abandon of a sign. An auditor may want to look into these cases to test whether these conditions are met or not. Also in patterns 3 and 4, but also in pattern 5, there is no Goods Receipt document entered into the ERP system. This has to do with the nature of the purchase order, whether it concerns goods or services. In case of services delivered, the goods receipt indicator is supposed to be flagged off and the GR activity may be abandoned. Again, an auditor may be interested to test these assertions. In the last pattern, the IR and GR have switched in following order. As depicted in Figure 1, these activities are allowed to appear in parallel order.

To examine all patterns requires too many recourses (recall that up to 390 events could be involved in one single audit trail). Therefore, we continue with a process discovery algorithm to reveal the sequence of activities within these patterns.

A process discovery algorithm extracts the executed process from an event log. As a start, we apply the fuzzy miner algorithm of [5] using default settings. This algorithm deals with the typical issues of real life data (completeness and noise) and simplifies and properly visualizes complex processes. The output is depicted in Figure 2. The thicker the line, the more frequent this sequence of activities occurs. The core process corresponds greatly to the designed process.

Table 2. Most frequent patterns

Pattern Sequence	Occurrences		Total	Throughput time (days)			
	#	%	%	avg	min	max	stdev
1 Create PO - Sign - Release - GR - IR - Pay	11 608	44.3%	44%	27.78	1	334	20.05
2 Create PO - Change Line - Sign - Release - GR - IR - Pay	6 955	26.6%	71%	32.33	2	343	57.72
3 Create PO - Change Line - Release - IR - Pay	2 488	9.5%	**80%**	75.63	3	344	38.99
4 Create PO - Release - IR - Pay	640	2.4%	83%	16.8	3	338	26.38
5 Create PO - Change Line - Sign - Release - IR - Pay	491	1.9%	85%	50.85	6	237	24.07
6 Create PO - Change Line - Sign - Release - IR - GR - Pay	393	1.5%	86%	56.36	9	295	40.16

The deviations are a Change Line that often occurs between the creation of the PO and the sign, and the existence of loops on every activity but the creation. Also between the payment and the invoice receipt there is some interaction. This result is discussed with the business expert and is considered to be normal. Using the default settings in this analysis reveals the general, more simplified, process followed in the event log. To uncover also less frequent followed paths, we set lower thresholds of the metrics, resulting in the model in Figure 3. Regarding Figure 3, we see immediately a more complex process with extra flows (edges). However, we have to be careful with interpreting these extra flows, because there could be an AND or OR relationship behind an edge. For instance, there could be a particular flow like Sign - IR depicted while in fact this is part of an AND relationship like 'after Sign: Release AND IR occur'. Further, some extra flows do not per se provide any problems, but should be examined further in a verification phase. In order to identify flows that require further examination, we specifically check (with a Linear Temporal Logic algorithm) whether the extra flows depicted in Figure 3 also really prevail in this order. This check is performed on the whole population set, again using the meta-data on activities and timestamps. The results of the checks are summarized in Table 3. Notice that these flows are subsets from complete audit trails, such as the ones in Table 2. The flows in Table 3 do not represent a complete pattern.

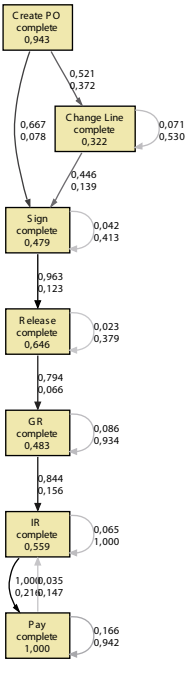


Fig. 2. Fuzzy Miner - default

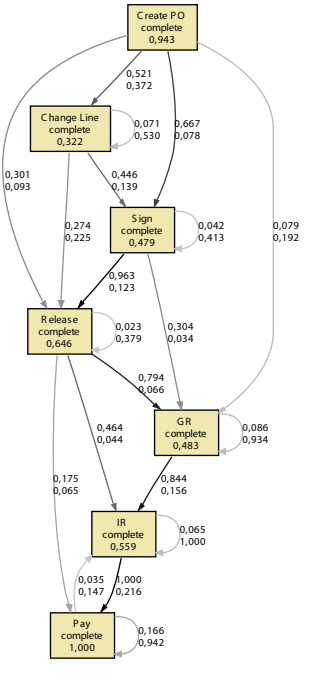


Fig. 3. Fuzzy Miner - lower thresholds

We find that the flow Create PO - GR does not explicitly occur in the data set, the other flows do. Flow 2 and 3 (occurring respectively 739 and 2 790 times) miss a sign. At the case company, there are situations where a release alone is enough as approval, but there are conditions to be met (maximum amount and specific document type). Whether these conditions were met in these cases will be examined in the verification phase. There are 11 cases where a sign was immediately followed by a Goods Receipt. This is not according policy rules, where a GR can only take place after a release. However, a manual check cleared these cases for further investigation. It appeared that the sign activities, which occur at PO header level and not at the detailed process instance level, were all triggered by a change in another item line than the one of the process instance. The GR activity on the other hand is related to the process instance itself and was not associated with the sign that took place just before the GR. The prevalence of the flows Release → IR (4 973) and Release → Pay (244) stresses the importance of the Goods Receipt indicator. The case company allows to discard the GR activity, but the Goods Receipt indicator needs to be flagged off in that case. This will be examined during the verification phase. At last, flows 6 and 7 stress the importance of examining whether for there exists for each payment a corresponding invoice. This too will be checked in the verification phase.

Table 3. Results of explicit checks on the extra edges in the Fuzzy Miner output

1	Create PO → GR	0	OK
2	Create PO → Release	739	Verification required on leaving out Sign
3	Change Line → Release	2 790	Verification required on leaving out Sign
4	Sign → GR	11	Manual check → OK
5	Release → IR	4 973	Verification required on GR indicator
6	Release → Pay	244	Verification required on GR indicator and IR
7	Pay → IR	227	Verification required on IR

As a last, very rudimentary control, we checked whether each process instance has at least one release activity in its pattern. We find three cases out of the population of 26 185 cases where there is no release. Two times it concerns a process instance that is created by a batch file, was paid and later on reversed by a person. Nevertheless, it got through the system without any approval. The third one is created by a person and is paid, 3 999.09 euro. Manual examination of this case revealed that the approval has been taken place outside the SAP workflow. That is why it is not in this event log. The auditor has to judge to what extent he wishes to investigate this further. We did not have insights into whether this deviation was acceptable or not by the company.

Role Analysis. During the role analysis, persons are linked to the activities they execute. This type of analyses makes use of the meta-data on activities and originators. The process view in itself is not used in these analyses.

Since a person often executes several activities, one person can bear multiple roles. In the role analysis, these roles are examined. The most straightforward example of role analysis is the segregation of duties (SoD) check. In the procurement cycle in our case company, three types of SoD should be respected: the

sign and release activities that are executed after each other should be executed by two distinct persons; the same holds for the Goods and Invoice Receipts; and for the release and GR activities.
In an exploratory step one can look at the Originator-Task matrix, a matrix providing the number of times a person executes a particular activity. This way one can verify whether some persons execute a double role that should not be combined in one case. We find for example in the excerpt of the matrix, given in Table 4, that person '...1' exercises both the roles of someone who signs and who releases. Person '...4' executes both Goods Receipts and releases. No example of a person combining the GR and IR roles is found in the matrix. Notice that identifying persons with combined roles only highlights the necessity to investigate the SoD further; it does not present violations of the SoD principle at itself. As long as these persons do not combine these roles in dealing with one single case, there is no problem. Also, the total matrix is of an extensive length (272 originators) to manually examine. For these reasons, we turn to a conclusive test to check whether the three SoD principles are applied correctly.

Table 4. Excerpt of Originator-Task matrix

Originator	1. Create PO	2. Sign	3. Release	4. GR	5. IR	6. Pay	Change Line
...1	0	171	11	0	0	0	0
...2	0	0	0	0	280	310	0
...3	0	0	23	0	0	0	0
...4	0	0	42	42	0	0	0
...5	0	24	0	0	0	0	0
...6	152	0	0	189	0	0	204
...7	...	...	...	...	...	...	...

In order to test the SoD principles a Linear Temporal Logic tool is used to check case by case whether a certain assertion holds. The assertions tested are:

- When a sign occurs, the following release activity is executed by a distinct person.
- A GR and IR are entered by two distinct persons.
- A release is given by a distinct person from the GR.

The first assertion needs to be tested pairwise, since there can be multiple signs and releases for one process instance - even though this is not included in the designed process. For instance if a release takes place and then a line is changed, the next sign is allowed to be performed by the previous releaser.
After testing all 26 185 cases (not a sample) by running the appropriate algorithm which tests the assertion case by case, we can conclude that the first two assertions hold in the investigated event log. Concerning the third assertion, 175 violations were found. Close examination revealed that the 175 cases were triggered by only three persons. One person violated the SoD principle on GR and release 129 times, another person 42 times, and a third person four times. Apparently, the ERP settings were not configured as desired. Whether these violations were formal (meaning that some informal communication justified the break of formal norm) or real, is up to the auditors to investigate.

Verification Phase by Attribute Analyses. As became clear in Table 2 and 3, some flows between activities or some assertions need to be verified. This is done by means of attribute analyses in a verification phase. This category of analyses uses attributes that are stored in the event log as input. As mentioned before, an attribute may contain information on the process instance itself (what is the type of the PO the line item belongs to, what is the value of the line item, was the Goods Receipt indicator turned on, etc.) or on an activity the process instance is submitted to (what amount is booked at a payment activity, what is the reference number of a GR document, etc.). The analyses in this section are a direct response to the output of the activity patterns found in the primary analyses, reported in Table 2 and 3.

A first analysis compares the references of the payment activities with the references of the IR documents, to check whether there is an accompanying invoice for each booked payment. Both reference numbers of the payment and the invoice are stored in the event log as attributes. This test resulted in 46 incorrect process instances, encompassing 265 stand alone payments. One process instance -of considerable length- has 131 pay activities without a corresponding IR, another 75, and yet another 10. The remaining process instances only have one, two or three stand alone payments. There were 17 originators responsible for these payments. One of these originators is responsible for 216 out of the 265 payments. Two other originators have respectively 18 and 12 stand alone payments on their account. These payments are all sorted out manually to check whether the payments could have been based on a Subsequent Debit, which is an alternative document for a standard invoice. This seemed indeed to be the case with all payments. The question remains why all these bookings are based on this type of document instead of on a regular invoice. This outcome warrants further investigation.

A second analysis, also as a follow-up of the revealed patterns in a previous step, investigates the functioning of the Goods Receipt indicator. If this indicator is flagged on, the accompanying process instance must have a GR before it can get paid. We tested whether all cases without a GR had indeed a Goods Receipt indicator that was turned off. There were three cases where this assertion did not hold, indicating a breach in the configuration settings of the ERP system. It would be interesting, in this context, to have an attribute on whether this case refers to services or goods. This kind of information was however not available. An auditor could judge whether or not to request the capturing of this data in the future.

The last analysis verifies whether the internal conditions of the organization are met when there is no sign in the activity pattern. For reasons of confidentiality we cannot give further details on this test, except that we have examined the document type and the amount of these process instances. 742 cases (2.8%), violating the preset conditions, were identified. Possible more exceptions than inserted in our check are allowed on this rule. The domain experts should however evaluate these rules and the settings.

Social Networks. Having the event log structured in the way required for process mining, allows us to construct social networks of employees involved in the process. This type of analyses reveals yet another powerful advantage of process mining: the ability to uncover collusion, a type of fraud known for its difficulty to detect. We built two social networks for subgroups of employees that drew our attention as a result of previous analyses. As was revealed during the role analysis, there is a subgroup of 175 cases where the SoD principle concerning the activities 'release' and 'Goods Receipt' was violated by three persons. In total, 24 originators were involved (this includes all activities instead of only the release and Goods receipt). The social network of these 24 originators, in these 175 cases, is depicted in Figure 4. As can be seen, there are three clusters of persons. The three persons violating the segregation of duty all take a central position in one of these clusters. This map of social network provides the opportunity to compare the designed organizational structure with the actual network. Another interesting subgroup to visualize a social network of is the group of 742 cases where no sign was present and the conditions for this exception were not met. This network -depicted in Figure 5- can yield valuable insights for the domain expert when provided with the names of the employees involved. We see that there are three groups of people. Two groups are connected to each other by two common persons. The third group is completely isolated.

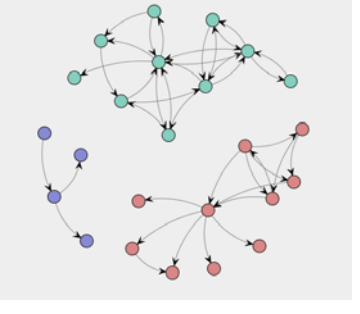


Fig. 4. Social network in which SoD Release-Goods Receipt is violated

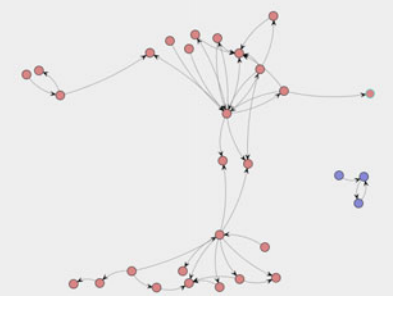


Fig. 5. Social network of 742 cases without signature

6.3 Summary of Results

Throughout the four types of analyses, the following issues that warrant further investigation were identified:

- Three PO's (out of the 26 185) passed the internal control system without any sign or release.
- 175 violations of the segregation of duty principle concerning Goods Receipt and release took place (filed by three persons).

- 265 payments (out of the 31 817) did not relate to an invoice, but to an alternative document type. These payments were concentrated in 46 process instances.
- Three PO's did not show a Goods Receipt entry in the system, although the Goods Receipt indicator was flagged.
- 742 cases did not show a sign activity, though the conditions for this exception were not met.

These issues were only identified using the process mining approach, and not during the traditional internal audit. During the traditional internal audit and accompanying internal control evaluation no internal control issues were identified; the ERP configuration settings were found to be strong internal control systems. This is not unexpectedly, given the small fraction of cases deviating from normal procedures identified during the process mining approach. Chances to uncover these cases when not testing the whole population are slim. We attribute the additional identified issues, resulting from the process mining approach, to three explicit aspects of process mining:

- The inclusion of meta-data which allows us to construct the real process execution, leading to more insights in the process and on its turn giving input for the verification phase.
- The process-view, which also led to more identified internal control issues in the studies of [6,7].
- The analysis of the population instead of a sample of the population.

7 Conclusions

In this report we present a first experience of employing process mining techniques in the context of internal auditing. The procurement process of a large multinational bank was analyzed. In line with a previous study on the use of monitoring technology (Masli et al. 2010) and other studies concerning a process-view approach like [7,6], we identify additional internal control issues with the process mining approach compared to the internal auditors' evaluation. Issues we identified all involved exceptions on the designed process, in a very small frequency of occurrence. The fact that these cases went through the internal control system is seen as evidence that control monitoring is never superfluous, even with the implementation of an ERP system.

Although process mining might be researched more intensively in other fields, this case study is the first attempt to bridge the gap between auditing and process mining. The incorporation of log data concerning user ids and timestamps of activities into the event log makes process mining a monitoring approach with groundbreaking possibilities. Given the right conditions, a process mining approach gives the opportunity to replay all cases that were part of a process and to analyze them fully and objectively, hereby representing the ultimate assurance tool.

References

1. Masli, A., Peters, G.F., Richardson, V.J., Sanchez, J.M.: Examining the potential benefits of internal control monitoring technology. Accounting Review 85(3), 1001–1034
2. van der Aalst, W.: Business process management center (2011), http://bpmcenter.org/
3. van der Aalst, W.M.P., Reijers, H.A., Weijters, A.J.M.M., van Dongen, B.F., Alves de Medeiros, A.K., Song, M., Verbeek, H.M.W.: Business process mining: An industrial application. Information Systems 32(5), 713–732
4. van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128–1142
5. Günther, C.W., van der Aalst, W.M.P.: Fuzzy mining – adaptive process simplification based on multi-perspective metrics. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 328–343. Springer, Heidelberg (2007)
6. Kopp, L.S., O'Donnell, E.: The influence of a business-process focus on category knowledge and internal control evaluation. Accounting Organizations and Society 30(5), 423–434
7. Bierstaker, J.L., Hunton, J.E., Thibodeau, J.C.: Do client-prepared internal control documentation and business process flowcharts help or hinder an auditor's ability to identify missing controls? Auditing: A Journal of Practice and Theory 28(1), 79–94

BPM Governance: An Exploratory Study in Public Organizations

André Felipe Lemos Santana, Carina Frota Alves, Higor Ricardo Monteiro Santos, and Adelnei de Lima Cavalcanti Felix

Center of Informatics – Federal University of Pernambuco
{afls2,cfa,hrms,alcf}@cin.ufpe.br

Abstract. Business Process Management is a widely known approach focused on aligning processes of an organization in order to achieve improved efficiency and client satisfaction. Governance is an important requirement to enable successful BPM initiatives. This paper provides a qualitative empirical study to investigate what BPM governance elements are adopted by teams conducting early BPM initiatives in public organizations. The results suggest that early BPM adopters in public sector face several barriers due to difficulties in acquiring professionals with BPM expertise, bureaucracy and legislation rigidity, among others. In particular, committed sponsorship and monitoring were appointed as important BPM governance facilitators by participants of the study. Findings also show that further empirical studies are needed to increase the body of evidence in this field.

Keywords: BPM governance, early BPM initiatives, public organizations, empirical qualitative study.

1 Introduction

Business Process Management (BPM) is a managerial approach that has been receiving growing interest from academy and industry in the last decade. According to Korhonen [1], BPM is a key paradigm of enterprise computing to increase agility in organizations. It has been considered a top priority for organizations trying to survive in highly competitive markets [2]. BPM refers to the management of the entire business process lifecycle, which includes: design, analysis, implementation, execution and continuous improvement of an organization's processes. It is a multidisciplinary field that integrates knowledge and practices coming from management and information systems disciplines. Organizations willing to start a new BPM initiative must build on a culture of change, continuous improvement and cross-functional team work [3].

Business Process Governance is frequently cited as a critical factor for the success of BPM initiatives [1],[2],[4],[5],[6],[7]. Loosely speaking, business process governance "governs" BPM, and its main purpose is to ensure that BPM delivers efficient results [8]. The resulting governance processes provide a reference framework to guide organizational units of an enterprise to ensure responsibility and accountability for adhering to the BPM approach. BPM governance can be considered the "definition" layer of BPM. According to Bandara [2], governance provides principles that

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 46–60, 2011.

support BPM initiatives by addressing ownership and control of process across organizational units and minimizing gaps between organizational strategy and BPM efforts. According to Rosemann and de Bruin [9], governance is one of key factors to build BPM maturity. The other factors are: strategy, culture, people/resources, IT and methods. Good governance is necessary for the success of business processes, which in turn, contribute to business success [4]. In public organizations, corporate governance is increasingly seemed as a key element to increase transparency, integrity and accountability in order to reinforce public confidence in government activities.

Despite several works raising the importance of process governance, there has been little progress on governance issues in the early stages of BPM adoption. In particular, there is a lack of empirical studies with methodological rigor to build a reliable body of evidence in this field. Evidence based studies are recommended to leverage the quality and confidence of research results [10]. This paper presents an exploratory field study to investigate what are the key governance elements for public organizations initiating BPM projects. In particular, our goal is to investigate the importance of governance elements for public organizations at the initial BPM maturity stage (i.e. maturity level 1) [9]. Our underlying assumption is that although immature BPM teams may not follow a formal governance model, they must adopt informal governance elements to enable improvement of their initiatives. Based on this assumption, our study aims to answer the following research questions:

> RQ1: What elements of BPM governance are employed in the public organizations?
> RQ2: What facilitators and barriers to BPM governance are found in the public organizations?

This paper is structured as follows: Section 2 presents definitions and elements of BPM governance. Section 3 presents the research method used to conduct the field study. Section 4 describes study results. Section 5 presents a discussion of findings and limitations of this study. Section 6 discusses related work. Finally, Section 7 concludes the paper and provides directions for future research.

2 BPM Governance

Literature presents several definitions for BPM governance, which has been also referred as business process governance, as can be found in [4], [5], [8], [11], [12], [13], [14]. To the purpose of bringing a theoretical foundation for our study, a useful definition is given by Kirchmer [8]. This author defines BPM governance as a set of guidelines and processes focused on organizing all BPM activities and initiatives of an organization in order to manage the BPM project.

BPM initiatives are not implemented in isolation. Instead, they are an integral part of overall business strategy and technology management. Therefore, as suggested by Khusidman [15], it is crucial to consider BPM Governance as an interoperable part within the "Ecosystem of Governances" of the organization. According [15], all categories of governance, such as: Process Governance, IT Governance, Business Service Governance, among others within the enterprise would follow similar approaches. This integrated viewpoint of all governance initiatives within an organization may increase efficiency of new ventures.

Many BPM governance elements can be found in literature. We present a synthesis based on authors Jeston e Nelis [5]; Korhonen [1]; Harmon [1], [16], [17]; Rosemann [9]; Richardson [13]; Kirchmer [8]; Spanyi [12]; Paim [18]; and Barros [14]:

- **Objectives:** refers to the objectives of BPM efforts in the organization. Usually process governance is concerned with ensuring the alignment of BPM initiatives to organizational strategic objectives.
- **Roles and Responsibilities:** these elements constitute the way people can act in a process with some authority, scope of activities and expected results. According to CBOK [19], it can be found in industry more than a hundred role names associated to business process management. "Business process owner", "business process manager", "business process analyst" and "business process consultant" are some illustrative examples. Governance of roles and responsibilities in business processes is very important to establish an adequate scope of tasks and reward system for BPM teams.
- **Standards:** includes many factors based on reference models. Standardization enables uniformity of business process initiatives, such as, methods, tools, metrics, process architecture and document templates. Many proposed standards in industry can also be used in BPM initiatives (e.g. Six Sigma, Lean). The governance of these factors is important in many ways: to create a common vision and language for BPM efforts, to improve communication, to facilitate the sharing of knowledge, and assess return of investment.
- **Tasks:** relates to actions necessary to execute BPM as an approach of organizational management. Design and model processes, monitor and report processes performance, inspect and audit processes execution are some examples of tasks conducted by different team roles.
- **Organizational Governance Structure:** relates to organizational structure of roles and teams involved at strategic, tactical and operational levels. This aspect is important to give sponsorship and empowerment to BPM teams. A Business Processes Office (or Business Process Center of Excellence), a Steering Process Committee and Process Project Teams are organizational structure elements cited in most process governance models.
- **Control Mechanisms:** involves control of how well the governance principles are being effective and in what level BPM initiative is in compliance with the process governance model in use. Inspection and audits may be listed as common control mechanisms. Control is at heart of governance to support corrective actions for continuous improvement and evolution of the governance model in use.
- **Assessment Mechanisms:** how to assess team performance regarding their contribution to achieving BPM objectives. This involves the creation and maintenance of a reward system in order to motivate individuals to work by "end-to-end" corporative process that aggregate value to clients, rather than just working inside the limits of their functional unities.

These governance elements should not be viewed as isolated items. For example: objectives, roles, responsibilities, and tasks have several interdependencies among them. Nevertheless, it is outside the scope of this paper to explicitly elicit all these interactions. In addition, we do not aim to formulate a precise conceptualization of

what constitutes a BPM governance model. In the field study, we did not follow a formal governance model. The reason behind this decision is because, by anecdotal evidence, we had the basic assumption that participants of organizations studied had limited awareness of BPM governance formal models. Instead, a synthesis of above mentioned BPM governance elements was used to guide us through the study in order to answer the stated research questions.

3 Research Method

As presented in Section 1, our research aimed to identify the importance of BPM governance at public organizations in early BPM projects. Given the exploratory and social-centric nature of this research, our field study followed a qualitative research approach. The qualitative approach allows the acquisition of novel knowledge and insights from empirical data. A non-probabilistic and purposeful sample selection strategy of organizations and participants was adopted, as recommended by [20]. We prioritized richness of the cases as basic selection criteria. Then a chain strategy [20, 21] was used, in which initial key participants suggested other cases and subject candidates. As a wide range of projects with different scope of activities could be seen as BPM initiatives, this study selected organizations undergoing process modeling activities with plans of conducting improvement actions in near future. Five organizations were initially invited to take part on the study. Finally, four organizations effectively accepted to participate. A brief description of the organizations is presented as follows:

- **Organization A[1]:** focuses on human resources, acquisitions and contracts administration of a Federative State in Brazil. Its clients are all the other governmental unities of this State (around 70 organizations). All study participants were from an organizational unity responsible for designing and managing human resources processes. Participants included: one BPM team leader, two process analysts, one Human Resources functional manager and one departmental chief (considered BPM sponsor).
- **Organization B:** supports educational services and policies. Its main clients are adolescent students. Participants were from an organizational unity team working on IT services, process modeling and design, and human resource management department. This team was working mainly on improvement of human resources management processes. Participants in our research included: two process analysts (one of them was an IT consultant who acted as part-time process analyst), one Human Resources functional manager (who was BPM client) and one departmental IT services chief (who acts as BPM sponsor).
- **Organization C:** works on social services and human rights policies. Its clients are population in general, particularly people with social support needs. In this study, participants were mainly from two organizational unities responsible for IT services, and special services for people with physical limitations. The participants were: two IT consultants who acted as part-time process

[1] As agreed with participants, all organization names are being omitted.

analysts, one departmental IT services chief (who acted as BPM sponsor), two functional managers (who were BPM clients) and one director (who was BPM client).

- **Organization D:** works on tourism policies and services development. Its clients are local population, tourists and enterprises involved in tourism industry. Participants involved were: two process analysts, one planning director (who acted as BPM sponsor) and two legal advisors (from a legal department, acted as BPM clients).

In summary, 19 professionals from four public organizations participated in the study. These professionals played the following BPM roles: process analyst, BPM team leader, BPM sponsor, BPM client. It is worth noting that several of these roles were informally described at the organizations.

Data was collected in two phases: initially, 13 interviews were conducted with 15 participants. Most interviews were individual, only two interviews had two participants. The interviews were semi-structured. This type of data collection strategy allows a flexible set of questions, and during the interview it is possible to improvise to explore emergent topics of interest [20], [21]. The interview guide had questions that expanded RQ1 (see Section 1) exploring what each BPM governance elements presented in Section 2 were applied by the organizations. In this phase, we adopted the basic assumption that, because participants were early BPM adopters, they were not necessarily familiar with the business process governance terms. However, we believe that they would have concerns in applying some BPM governance concepts, possibly in an informal way. These assumptions were then confirmed during the interviews. To avoid misunderstandings, governance concepts were presented in indirect manners. A copy of interview guide can be obtained by demand to us.

All interviews were audio recorded and transcribed. Two authors transcribed the interviews separately, and then the other two authors validated the results. We adopted an open and axial coding strategy to generate categories. A spreadsheet was created with interviews text and generated categories. This process was conducted in an iterative fashion, where several meetings were arranged to consolidate the results.

In the second phase, a focal group meeting was conducted with twelve participants. Several interviewees also participated in the focal group. In the first part of the meeting, one of the authors acted as facilitator and presented the main concepts of BPM governance in order to inform the group and stimulate initial discussions. Then, elements of BPM governance were presented and participants were asked to prioritize the elements based on their perception of importance. After that, the facilitator asked if participants were aware of other governance elements. Finally, participants were asked to describe the main facilitators and barriers to their BPM initiatives. The main objective of this second phase of data collection was to validate the findings obtained from individual interviews. We also tried to explore open issues that were not sufficiently treated by the interviews. Our effort to minimize interpretation bias was addressed by each author analyzing and categorizing data separately and holding discussion sessions to aggregate collected evidence. We also used two data collection methods (i.e. interview and focal group) in order to increase the strength of evidence.

4 Results

The results of the study show that all four BPM initiatives have begun around one year or less prior the study. Therefore, we can consider that all organizations are immature in terms of BPM expertise. In particular, people involved have not been properly trained on BPM skills. In organization B, workflow automation was done in some processes for human resource management. The organization hired external consultants to support process elicitation, design and automation. Our investigation has also evidenced that in organizations A, C and D, the BPM initiative can be characterized as a wide effort covering several organizational units. This means that the BPM scope covers large inter-departmental processes and has sponsorship from the executive managers. Following, we present evidence to answer the research questions.

4.1 (RQ1) What Elements of the BPM Governance Are Employed in the Public Organization?

It was observed that interviewees were not familiar with the term "process governance". In particular, there was not any explicit process governance model in operation. For participants with IT background, governance was not properly a new idea because of previous knowledge in IT governance, but they did not know the concept would also be applied for BPM. In spite of that, concerns with some process governance elements were encountered in an implicit way on interviewee's discourse and actions. For example, in organization A, a standard procedure for business process modeling and monitoring was developed as way to guide process analysts' work. In organization D, document templates for process diagnosis were created. These results suggest that concerns regarding process governance importance do exist at the organizations studied, but none of them follow a formal BPM governance model. In the next sections, we explore what BPM governance elements are being employed by studied organizations.

4.1.1 Roles and Responsibilities

We found the following BPM roles and respective responsibilities in the studied organizations:

- **Process Analyst:** responsibilities involve the design and analysis of processes. Implementation of changes was also appointed in the scope of responsibilities. However, at the time of the study most projects were not yet at the phases of deploying "to-be" processes. In organizations B and C, this role was played part-time by IT analysts. In general, the role was not formally defined in the majority of BPM initiatives studied.
- **Process Project Manager:** responsible for planning and monitoring process projects. In most cases, this role was played in conjunction with "process analyst" role by a single person. Only in organization A, this role was fully assigned by one individual.
- **Functional Process Manager:** responsible for managing processes in departmental units. People responsible for this role were the main source of information for process analysts conducting design activities. End-to-end process manager

was not found in any organization. However, this particular role is always suggested as necessary in BPM literature.

- **Process Project Sponsor:** a role widely reported in project management literature [22]. The sponsor is responsible to perform executive support (budget, decision making, priority definitions and maintenance). In participating organizations, this role did exist but it was assigned to several senior managers in an informal way.

We observed that in all organizations these roles were not formally defined, nor responsibilities clearly specified. When inquired about it, interviewees just talked about the activities they were assigned, but they did not have a critical position regarding this issue. It seems that in most cases it was not a common task defining roles and responsibilities. In organizations C and D, participants reported difficulties to define roles and responsibilities. This was mainly related to problems of weak organizational structure and lack of personnel. As described by participants during the focal group session:

> *"We have problems defining staff roles and responsibilities, this is related to lack of proper organizational structure. Unities in our organizational structure are not well defined as our organization is very new. Besides that, there is a high turnover of staff".* Anna[2], process analyst.
> *"In my case there is lack of personnel available. My boss cannot allocate people in well defined positions. People have to do more things than what is planned in our organizational structure".* Claire, functional process manager.

These findings reflect the historical structure of public sector organizations in Brazil. In particular, the absence of staff performance metrics as well as changing managerial priorities increases the difficulty to define roles and responsibilities.

4.1.2 Standards

We found that organizations mainly adopted BPMN (Business Process Modeling Notation) notation [23] and *Bizagi*[3] tool to support process design. In most cases, the tool adoption was very recent. We noted that staff did not have proper BPM training. Most of them learned concepts and notations informally. In Organization A, we identified an effort to create a standard procedure to conduct process design, analysis and monitoring. However, the initiative was very recent and considered immature to be fully adopted. In most cases, standard procedures and document templates were not identified. It was evidenced that process management team had insufficient experience and knowledge on BPM tools and methods, especially in process automation. As participants mentioned in the focal group:

> *"I am worried about implementation of processes in terms of systems and tools. We don't have proper training; we just try doing it as we can".* Claire, functional process manager.

[2] All participants names are fictional, as agreed with interviewees.
[3] Process modeling and execution tool available at www.bizagi.com

4.1.3 Tasks

The main process management tasks performed by teams were process modeling and documentation. In organizations A, C and D, implementation of changes and automation were not done yet. As mentioned in Section 4.1.1, teams did not formally delimit tasks and responsibilities assigned to particular roles. There was strong evidence that proper training, knowledge and experience in BPM tasks need to be improved.

4.1.4 Structure for Governance

In all studied organizations, there was not a formalized BPM governance model. In spite of finding many implicit concerns about BPM governance elements in participant´s discourse, senior managers did not endorse an enterprise-wide BPM governance model. In organizations B and C, there was not a process dedicated team. While in organizations A and D, there was a BPM team. In particular, organization A was the unique case in which the BPM team was formally defined in their organizational structure. In that case, the team had been formed recently, and the professionals were still not experienced in BPM, because they were recruited for the generic position of *Management Analyst*.

4.1.5 BPM Goals

According to participants in all organizations, the goals of their BPM projects were well known. *Improvement*, *standardization* and *normalization* were the key goals reported. In particular, *alignment* and *integration* of processes from different organizational units, and *process automation with BPMS* were also examples of goals reported. In all organizations, participants claimed that project goals were aligned with executive sponsors expectations. However, BPM project goals were not formally associated to organizations strategic plans. This means that goals were not explicitly shared by all stakeholders. As results from the focal group, we observed that:

- Organizations face difficulties to establish and maintain clear priorities by executive management *"Priorities are for very short term. There is no long term plan. Problems repeat"*, Claudius, process project manager.
- In some cases, the lack of human resources cause accumulation of work for the team who have to accomplish multiple tasks *"We can never be focused on a single demand, we have always 'hundreds' of demands to respond",* Claire, functional process manager.
- Government shift was mentioned as a key threat to BPM initiatives *"in public administration we suffer with that: change in government may change initiatives. There should have continuity of plans independently of who comes and who goes"*, Sally, functional process manager.

4.1.6 Control Mechanisms

Due to the fact that formal BPM governance models were not adopted, we did not find formal control mechanisms to governance elements considered in this study. However, the majority of interviewees reported that control meetings were periodically done to assess project progress. This result suggests that low maturity BPM

teams should adopt a set of policies in order to obtain better control of their initiatives. A suitable control mechanism could involve a balancing effort among compliance obligations, business objectives and risks. Control is particularly important in the context of public organizations because society expects increased transparency and performance of public actions.

4.1.7 Assessment Mechanisms

There was no evidence gathered in the study of any assessment mechanism. We did not find mechanisms to evaluate staff performance at studied organizations. In spite of that, concerns with this aspect were described here: *"the remuneration and rewards should be reviewed"* (Daniel, process analyst). Some participants demonstrated they were skeptical with "*yet another process improvement initiative"* (Laura, functional manager). Employees with this viewpoint may be resistant to change. We believe that recognition and reward are important drivers to incentive early BPM teams.

4.1.8 Other Potential Governance Elements

The previous BPM governance elements were collected in our literature review. Other potential governance element were revealed in our study:

- **Legislation and regulations for public sector:** given that all studied organizations are from public sector, legislation was considered a very important issue for governance. In organizations' country, actions conducted by government institutions have to be strongly supported by legal issues in a way that the set of possible legal acts turns to be more restricted then the one of private sector. In Private sector organizations, legal acts are everything that is not displayed as prohibited by laws and regulations. While, in public sector organizations, all the possible legal acts have to be previously and explicitly displayed by laws and regulations, and everything other than this is prohibited. Legislation and regulations can be used as important impersonal mechanisms to enforce BPM Governance, as suggested by Markus and Jacobson [4]. However, outdated or improper legislation and regulations may be also a serious barrier to effective BPM governance.

4.2 (RQ2) What Facilitators and Barriers to BPM Governance Are Found in the Public Organizations?

Tables 1 and 2 present the main facilitators and barriers to process governance revealed from our study. They were obtained from categories identified during analysis of interview transcripts. The facilitators and barriers are ordered by frequency of occurrence in participants discourse. However, we do not claim that this means order of importance. A further prioritization work, possibly with multi-criteria decision making methods shall be conducted to obtain that result.

All facilitators were related to managerial and social factors. IT aspects were not reported as key enablers of BPM. At the current stage of their projects, participants

were not concerned with technology, they needed to address organization challenges first. It is worth noting the list of facilitators is significantly smaller than list of barriers. That is probably due to the fact that all BPM initiatives studied are still in early phases, not sufficiently stabilized and without substantial results.

Table 1. Facilitators to BPM Governance

Facilitators	Evidence from field study
Sponsorship from Governor and Secretaries	*Process Analyst "There is a real intention to improve processes. Actions from governor helps initiatives, even though people resist changes, they know orders come from the top"*
BPM initiative monitored by sponsors	*BPM leader "Actions from our current sponsor have been very effective. He participates of meetings, and follows the evolution of the project".*
Support from other governmental organizations	*Internal client "Resources are available; there is a strong credibility of our government from funding organizations. Our collaboration with other secretaries helps the success of our project".*
Process team formed by internal staff	*BPM leader "process initiatives have started from internal staff".*
Qualified and motivated professionals	*Process Analyst "Hire experienced public managers, analysts... Our sponsor has large experience, and gets involved in the projects".*
Cooperation with process clients	*Process Analyst "The areas are receptive. They need someone to support them to model the process and provide recommendations."*

Table 2a. Barriers to BPM Governance

Barriers	Evidence from field study
Change resistance	*Process Analyst "We found several difficulties involving people not willing to pass information to model the processes."*
Inexperience in BPM Approaches	*Process Analyst "Given that it is our first BPM Project we don´t have a clear vision on how to do things."*
Lack of methods	*Process Analyst "There is no standard methodology. We ask people to contribute, to participate in meetings, to help modeling processes. "*

We observed that the group of categories formed by *inexperience in BPM approaches*, *lack of methods* and *lack of BPM training* are powerful barriers that need to be treated urgently if these organizations want to succeed in their BPM initiatives. As participants pointed that they have s*ponsorship from Governor and Secretaries*, it can be argued that they should start planning how to invest to overcome these barriers.

Table 2b. Barriers to BPM Governance (continuation)

Lack of BPM Training	*Process Analyst "We have difficulties in visualizing how to do process automation, because we don't have specific training for it".*
Internal clients with poor experience in basic IT tools	*Process Analyst "Our internal clients have difficulties to use simple Technologies, such as using emails, imagine how these users are going to operate a BPMS to authorize internal travels."*
Bad experience with previous external BPM consulting	*BPM leader "We have introductory meetings to explain clients how our work is conducted, they replied saying that other consultants came here asking questions but they did not contribute anything to support our work."*
Bureaucracy and slow governmental processes	*Internal Client "The biggest challenge is to ensure that process analysts will be able to break the current paradigm from public sector."*
Problems with current legislation	*Process Analyst "Some of our actions are blocked by legislation."*
Inadequate reward system	*Process Analyst "If we want to work with process management focusing on improved results, we must have a financial reward to ensure better performance."*
Internal clients with precarious infra-structure (physical space, equipment, technology)	*Process Analyst "Our main difficulty is structure, it is too precarious, physical space, equipment and limited number of staff."*
High team turnover	*Process Analyst "Turnover is very high here. In average, 80% of staff are from outsourcing companies."*
Low integration among governmental organizations	*Process Analyst "There is a lack of integration among secretaries, this leads to a lot of rework."*

5 Discussion

5.1 Implications for Research and Practice

This empirical study has several implications for research and practice in the field of BPM governance. Before conducting the field study, two authors had some previous knowledge of the BPM efforts of the four organizations. Therefore, we had the initial assumption that participants had limited awareness of BPM governance terminology and formal models. However, regarding the governance elements that guided our study, participants recognized the importance of several elements, such as: *roles*, *tasks*, *standards*, *objectives* and *structure*. Other elements proposed were not perceived as important, such as: *assessment* and *control mechanisms*. These elements were not emphasized during interviews and discussions. It may be due to the fact that teams had limited experience and their initiatives were at very initial stage. It may also indicate that some governance elements are more important than others for

particular situations, for instance, public organizations and specific BPM lifecycle phases. An additional element that emerged from the study is *legislation*. As far as we are concerned it is not included in BPM governance models proposed in the literature. This suggests that process governance models should be sufficiently flexible to allow the inclusion/exclusion of new elements. There was evidence that the importance of these elements may vary depending on the type of organization (i.e. public or private), maturity level of BPM initiative and team expertise. Therefore, a BPM governance model must be situational.

Despite the lists of facilitators and barriers to BPM governance were not prioritized, we argue that team development and training in BPM principles, methods and governance are key expertise to early BPM adopters. Training is necessary in order to enlarge the team´s vision of what needs to be done, and improve their readiness to adopt BPM standard practices and models. BPM expertise must fit with organization culture and maturity. Investment in BPM consultancy is a recommended option. In addition, it seems to be very important to reinforce sponsorship of BPM initiatives. This aspect was pointed as a key facilitator by participants. They also emphasized the importance of having clear objectives and priorities explicitly assigned by executive managers and widely communicated to all teams. This suggests that governance is an important factor to be invested since the initial phases of BPM in order to sustain it in the long term.

A BPM maturity model [24], [9] may be helpful to guide the process improvement as they could give insights of "what to do next". We believe that further studies are needed to better investigate the interrelationship between governance and maturity of BPM initiatives. A key insight that emerged from our study is that without investment in BPM governance it is difficult to improve BPM maturity as a whole. Our study also showed that for public organizations, inter-organizational BPM governance aspects must be considered in order to guarantee the alignment and interoperability of processes from different public organizations. This is due to the fact that public organizations are interconnected in what we can call a government ecosystem.

5.2 Limitations

Due to the limited number of organizations studied and low maturity of their BPM initiatives, it is impossible to provide a definitive advice on the state of practice in BPM governance. Instead, this study provided an exploratory qualitative investigation in this field. A key limitation of this paper is eventual bias in the selection of organizations. We adopted a non-probabilistic and purposeful selection strategy, which we believe suited our needs for this initial study. To minimize interpretation bias, data was analyzed separately by each author, and then we aggregated viewpoints during discussion meetings. However, we often found that our interpretation was hindered by subjectivity in interviewees' speech. Therefore, there is a possibility that the analysis process may have resulted in some inaccuracy.

We adopted governance elements common to diverse authors work as basis to this study, instead of following a specific model or framework. Due to that, it is possible that some expected aspect will be missed by readers interested in some particular governance model.

We recognize that findings of this study may not be generalizable. However, providing a statistical quantitative study to generalize knowledge about BPM Governance in a certain population of organizations was not the intention of this study. In fact, the own nature of exploratory qualitative research does not seek to obtain generalized results [20]. In order to increase the body of evidence in the field of BPM governance, we encourage researchers and practitioners to conduct further empirical studies of the same phenomenon.

6 Related Work

Studies [3] and [25] are examples of empirical studies similar to this research. In [25], an empirical research was conducted in four South African Financial Services Organizations that made an investment in a BPM suite. The goal of the research was to make explicit the enablers of BPM success. In that study, they considered governance as a success enabler. Governance should cover the establishment of relevant and transparent accountability, decision-making and reward processes to guide individual's actions. In study [25], it was conducted a very similar study to ours in terms of research strategy and context. They proposed a BPM governance model that sets BPM decision-making, along with roles and responsibilities in an Australian governmental corporation. They adopted a qualitative case study strategy to analyze organizational documents using a content analysis approach and in-depth semi-structured interviews with key stakeholders. They also tried to address a gap in literature on how to deploy BPM Governance in practice. Differently from our study, the ones mentioned above were applied in organizations where BPM initiatives appear to be in a more mature level. The novelty of our study was exploring the importance of governance elements to BPM early adopters.

7 Conclusion

This paper aimed to build a cumulative body of evidence on BPM governance. The main contribution was an exploratory study conducted at public organizations. Initially, the paper presented an informal review in BPM governance. From our literature review, we observed that papers on business process governance lack details of empirical applications of governance models. Most proposed models do not provide implementation guidelines. It is also worth noting that most industrial papers are whitepapers with inadequate academic rigor. More specifically, papers do not clearly present research objectives and questions, methods, and limitations of the studies. These relevant gaps suggest that BPM governance should be investigated in a more rigorous way.

We conclude that an early introduction of BPM governance elements could guide immature BPM teams to increase success. Another important finding from our study is that we need to establish a research agenda for BPM governance. In particular, further empirical studies both in public and private organizations are needed. We plan to continue our studies with public organizations in Brazil. Following the research presented in this paper, we plan to conduct a collaborative intervention study based on

action research method. This new study will also involve participating organizations to explore and share experiences in their BPM projects. In particular, we aim to investigate what are the main inter-organizational business process governance issues to enable successful BPM initiatives in government ecosystem.

References

1. Korhonen, J.: On the Lookout for Organizational Effectiveness – Requisite Control Structure in BPM Governance. In: 1st International Workshop on BPM Governance – WoGo (2007)
2. Bandara, W., Indulska, M., Sadiq, S., Chong, S., Rosemann, M., Green, P.: Major Issues in Business Process Management: An Expert Perspective. Department technical report. School of Information Technology and Electrical Engineering, The University of Queensland (2007)
3. Thompson, G., Seymour, L.F., O'Donovan, B.: Towards a BPM success model: An analysis in south african financial services organisations. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) Enterprise, Business-Process and Information Systems Modeling. Lecture Notes in Business Information Processing, vol. 29, pp. 1–13. Springer, Heidelberg (2009)
4. Markus, M.L., Jacobson, D.D.: Business Process Governance. In: Brocke, J.v., Rosemann, M. (eds.) International Handbook on Business Process Management 2 - Strategic Alignment, Governance, People and Culture, pp. 201–223. Springer, Heidelberg (2010)
5. Jeston, J., Nelis, J.: Business Process Management – Practical Guidelines to Successful Implementations, 2nd edn. Elsevier/ Butterworth-Heinemann (2008)
6. Ravesteyn, P., Batenburg, R.: Surveying the critical success factors of BPM-systems implementation. Business Process Management Journal 16(03), 492–507 (2010)
7. Rosemann, M., Brocke, J.v.: The Six Core Elements of Business Process Management. In: Brocke, J.v., Rosemann, M. (eds.) International Handbook on Business Process Management 1 - Introduction, Methods, and Information Systems, pp. 107–122. Springer, Heidelberg (2010)
8. Kirchmer, M.: Business Process Governance for MPE. In: High Performance Through Process Excellence From Strategy to Operations, pp. 69–84. Springer, Heidelberg (2009)
9. Rosemann, M., de Bruin, T.: Towards a Business Process Management Maturity Model. In: Proceedings of the 13th European Conference on Information Systems, ECIS (2005)
10. Kitchenham, B., Dyba, T., Jorgensen, M.: Evidence-based software engineering. In: Proceedings of the 26th International Conference on Software Engineering, pp. 273–281 (2004)
11. Harmon, P.: Process Governance. BPTrends (2008), `http://www.bptrends.com/publicationfiles/advisor200802121.pdf`
12. Spanyi, A.: Business Process Management Governance. In: Brocke, J.v., Rosemann, M. (eds.) International Handbook on Business Process Management 2 - Strategic Alignment, Governance, People and Culture, pp. 223–238. Springer, Heidelberg (2010)
13. Richardson, C.: Process Governance Best Practices: Building a BPM Center of Excellence. BPTrends (2006)
14. Barros, D.B.: Governança de Processos: Proposição de um Modelo Teórico de Governança para Gestão de Processos. Dissertação de Mestrado. UFRJ/ COOPE/ Programa de Engenharia de Produção (2009)

15. Khusidman, V.: BPM Governance Framework. BPTrends (2010), http://www.bptrends.com/publicationfiles/ONE%202010-ART-BPM%20Governance%20Framework-VKhusidman-v5.pdf
16. Harmon, P.: BPM Governance. BPTrends 3(3) (2005a), http://www.bptrends.com/publicationfiles/bptemailadvisor020805.pdf
17. Harmon, P.: Best Practices in the Governance of Business Process Management. BPTrends, 1–23 (2005b), http://www.bptrends.com/publicationfiles/05-25-05BPT1HrTalkforIQPCBPMConf-Harmon.pdf
18. Paim, R., Nunes, V., Pinho, B., Santoro, F., Cappelli, C., Baião, F.A.: Structuring a process management center of excellence. In: Proceedings of the 2009 ACM symposium on Applied Computing - SAC 2009, p. 281. ACM Press, New York (2009), http://portal.acm.org/citation.cfm?doid=1529282.1529342
19. ABPMP: CBOK - Guide to the Business Process Management Common Body of Knowledge. Version 2.0 – Second Release, Association of Business Process Management Professionals – ABPMP (2009)
20. Merriam, S.B.: Qualitative Research – A Guide to Design and Implementation. Jossey-Bass, San Francisco (2009)
21. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14(2), 131–164 (2008), http://www.springerlink.com/index/10.1007/s10664-008-9102-8
22. PMI. A Guide to the Project Management Base of Knowledge (PMBOK® Guide). 4th edn., Project Management Institute (PMI) (2008)
23. OMG. Business Process Model and Notation (BPMN). Object Management Group–OMG, http://www.omg.org/spec/BPMN/
24. OMG. Business Process Maturity Model (BPMM) Version 1.0. Object Management Group – OMG (2008)
25. Doebeli, G., Fisher, R., Gapp, R., Sanzogni, L.: Achieving sustainable business process management and improvement through BPM governance Using BPM governance to align systems and practice. Emerald, Business Process Management Journal (2011) (pre-print)

Evaluation of Cost Based Best Practices in Business Processes

Partha Sampath and Martin Wirsing

Institute of Computer Science,
Ludwig-Maximilians-University, Oettingenstr. 67, 80538 Munich, Germany
p.sampathkumaran@accenture.com, martin.wirsing@lmu.de
http://www.pst.ifi.lmu.de/

Abstract. Reducing the Cost of a Business Process is a challenge faced by organizations. Business Process researchers have recommended a host of best practices for Business Process design which leads to Cost effectiveness. However, these are theoretical and there is no real guideline for either implementation or the Cost reduction acheived by the implementation of these best practices. In this paper, we evaluate the most commonly recommended best practices available in literature for Cost reduction for their effectiveness. We implement a pattern based Cost calculation methodology which shows the impact of best practices on examples in a measurable way. Using this methodology we calculate the overall Cost, reliability and the Cost incurred to achieve one successful execution of the Business Process; the Business Cost of the process.

Keywords: Business Process, Cost, Best Practices.

1 Introduction

In general profitability is the primary goal of any Business enterprise. The success of a Business is dependent on high income and low and controlled expenses. Processes are implemented to either directly contribute or support this goal of an organization. Every organizations interest is to make these processes successful. In the years, as information technology has become industry oriented the factors that make a process successful have taken center stage. This is especially very visible in the service industry. The aim of achieving higher quality and at the same time keeping the Costs controlled or reduced are of high importance to the Business. Even though this is important, methodologies, frameworks and theories which have foundational reasoning to achieve this are not precise in their recommendations. This is especially the case when we come to the topic of financial evaluation and optimization of a Business processes at the operational level.

In this study we consider five commonly recommended best practices: Resequencing of Tasks, Knock-Out Order, Task Elimination, Order Type and Triage, and Parallelism, for Cost optimization in Business Processes and evaluate them for their impact on a Business Process. So as to calculate the Costs before and after implementing these best practices we base ourselves on a pattern based Cost

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 61–74, 2011.

calculation methodology. We calculate, for each example, the Cost, Reliability and Business Cost i.e the Cost incurred in achieving one successful execution of the Business Process.

2 Pattern Based Cost Calculation

The methodology for Cost calculation of a Business Process based on repetitive patterns has been presented by Sampath and Wirsing [1]. The methodology considers each artifact within a Business Process which is represented as a Business Process Diagram and attaches a parameter for Cost and Reliability to the same. It then breaks the Business Process into repetitive patterns and the Cost. Further Reliability and Business Cost of each pattern is calculated. In turn the overall Cost, Reliability and Business Cost of the complete Business Process is generated. A single task with Cost C and Reliability R has the Business Cost as shown in equation 1.

$$BusinessCost = C/R \tag{1}$$

Four patterns are defined as basis for Cost calculations.

2.1 Pattern 1: n Tasks in a Sequential Order

This pattern considers n tasks, each having a Cost and reliability, in a sequential order as shown in the Fig. 1.

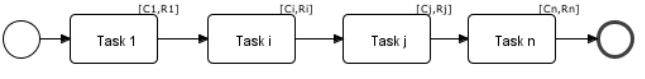


Fig. 1. n tasks in a sequential order

The calculation for Cost, Reliability and Business Cost is as shown here.

$$Cost = \sum Ci,\ where\ Ci\ is\ the\ cost\ of\ task\ i \tag{2}$$

$$Reliability = \prod Ri,\ where\ Ri\ is\ the\ reliability\ of\ task\ i \tag{3}$$

$$BusinessCost(1,n) = (Cn + BusinessCost(1,n-1))/Rn \tag{4}$$

2.2 Pattern 2: n Tasks in a Parallel Order

The pattern considers n tasks in a parallel order as shown in Fig. 2. The resulting cost and reliability of this parallel pattern then would be:

$$Cost = \Sigma Ci,\ (Ci\ is\ the\ cost\ of\ each\ flow\ in\ the\ parallel\ flow) \tag{5}$$

$$Reliability = Minimum(R) \tag{6}$$

$$BusinessCost = \Sigma BusinessCost(i)\ (i\ is\ the\ pattern) \tag{7}$$

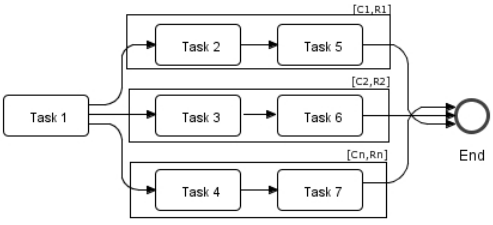


Fig. 2. n tasks in parallel order

2.3 Pattern 3: Conditional Branching

The pattern considers a conditional branching leading to different execution paths. The situation here is the same as mentioned in the case of sequential tasks in Pattern 1. Even though a probability has to be attached to each flow out of the Gateway. The corresponding cost of the path is then multiplied by the probability which will lead to the cost of the whole branching.

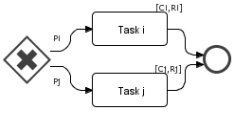


Fig. 3. BPD with conditional branching

$$Cost = \sum Pi * Cost(i),\ (Pi\ is\ the\ probability\ of\ taking\ path\ i) \tag{8}$$

$$Reliability = \sum PiRi,\ (Ri\ is\ the\ reliability\ of\ path\ i) \tag{9}$$

$$BusinessCost = \sum Pi * BusinessCost(i) \tag{10}$$

2.4 Pattern 4: "n" Successive Possibilities

The pattern considers n different services each performing the same function. The resultant parameters of Business Cost and reliability are dependent on the number of possibilities.

$$BusinessCost(1, Nn) = Cost(1, Nn)/Reliability(1, Nn) \tag{11}$$

$$Reliability(1, Rn) = 1 - ((1 - R1)(1 - R2)..(1 - Rn)) \tag{12}$$

3 Evaluation of Best Practices

In this section we consider five of the most commonly recommended best practices for Cost optimization. For each best practice, we take an example as an "Original Case", implement the recommended best practice on this example as a "Changed Case" and evaluate the impact of the same.

3.1 Resequencing of Tasks

This best practice is also called "Process Order Optimization"and is mentioned by Klein [2] and [3]. In a Business Process, the ordering of tasks does not reveal the logic behind the process and hence it could be that tasks in a process are executed even though it is not required at that moment. This best practice recommends that tasks such as these when resequenced in a Business Process help in Cost reductions. The logic is to execute a task only when the task is really needed to be executed.

Original Case. To evaluate this best practice, we consider a Business Process to book a flight. The corresponding BPD is as shown in the Fig. 4. The Business Process authenticates the customer, validates the inputs and finds the flights. If a flight is found then the flight is booked and confirmation is sent to the customer. We assume that the flight is available in 50 percent of the cases. Also, we make assumptions on the Cost and Reliability of the tasks in the Business Process.

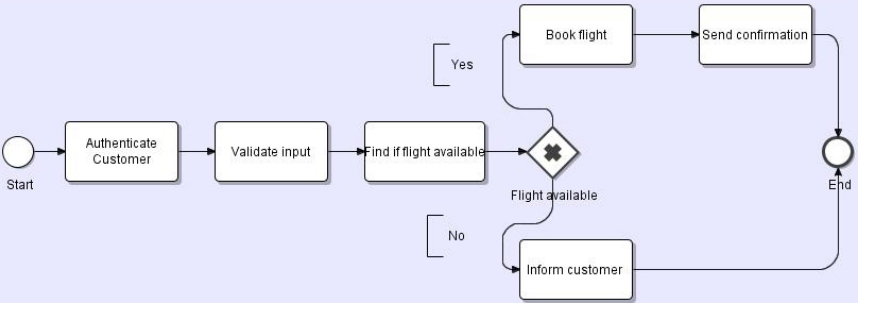


Fig. 4. Business Process Diagram to book a flight

Cost calculation: To calculate the Business Cost of this process, we divide the BPD in patterns which we combine by a decision (see Fig. 5). For each of the patterns, we make assumptions on the Cost and Reliability of all the tasks. In turn we calculate the Business Cost for each of the patterns.

The calculation at the level of the patterns is as shown in the table 1.

From the calculations in the table get the Business Cost of the patterns as:

$$Business\ Cost(Pattern\ 1) = 35.6$$
$$Business\ Cost(Pattern\ 2) = 48.1$$
$$Business\ Cost(Pattern\ 3) = 5.56$$

We assume that there is always a 50 percent chance of finding the flight according to the customer inputs. Hence the Business Cost of Pattern 2 and Pattern 3 would then be:

$$Business\ Cost(Pattern\ 2\ with\ Pattern\ 3) = 45.7 * 0.5 + 5.56 * 0.5 = 26.85$$

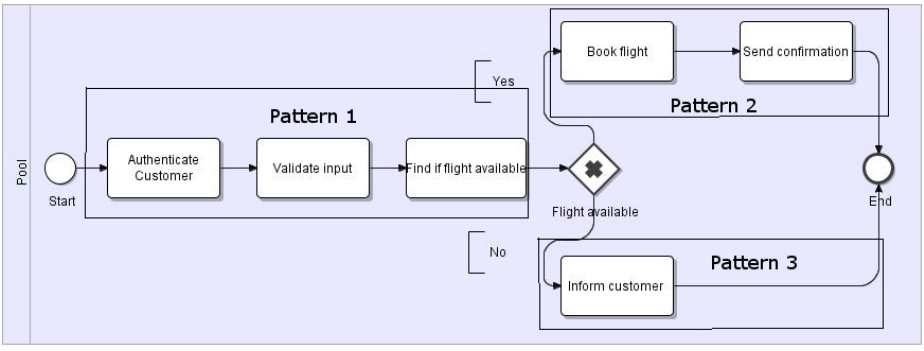


Fig. 5. Patterns in the Business Process Diagram

Table 1. Flight Booking - Pattern 1

Task	Cost	Reliability	Business Cost of the task	Business Cost
Pattern 1				
Authenticate	10	0.9	11.11	11.11
Validate input	10	0.9	11.11	23.5
Find if flight available	5	0.8	6.25	35.6
Pattern 2				
Book Flight	30	0.9	33.33	33.33
Send confirmation	10	0.9	11.11	48.1
Pattern 3				
Inform Customer	5	0.9	5.56	5.56

Table 2. Flight Booking - Pattern 1

Task	Cost	Reliability	Business Cost of the task	Business Cost
Validate input	10	0.9	11.11	11.11
Find if flight available	5	0.8	6.25	20.1

Hence the total Business Cost of the BPD would then be:

$$Business\ Cost(Pattern\ 1\ (Pattern\ 2\ with\ Pattern\ 3)) = 35.6 + 26.85 = 62.4$$

Changed Case with Tasks Resequencing. In the BPD in Fig. 4 we resequence the tasks in such a way that a task is executed only when it is required. The task "Authenticate Customer" is moved to the part where flight has already been found. This change is shown in Fig. 6 which is also shown in the patterns.

Tables 2, 3, 4 show the Cost and Reliability of the tasks and the patterns together.

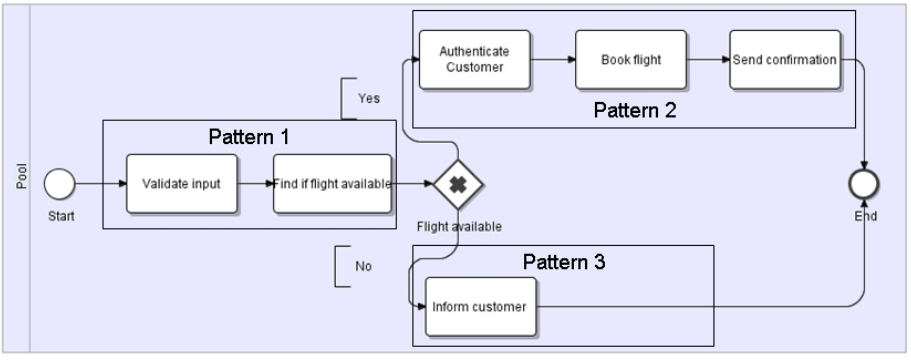


Fig. 6. Business Process Diagram to book a flight

Table 3. Flight Booking - Pattern 2

Task	Cost	Reliability	Business Cost of the task	Business Cost
Authenticate Customer	10	0.9	11.11	11.11
Book Flight	30	0.9	33.33	45.7
Send confirmation	10	0.9	11.11	61.9

Table 4. Flight Booking - Pattern 3

Task	Cost	Reliability	Business Cost of the task	Business Cost
Inform Customer	5	0.9	5.56	5.56

From the calculations in the tables we calculate the Business Cost as:

$$Business\ Cost(Pattern\ 1) = 20.1$$
$$Business\ Cost(Pattern\ 2) = 61.9$$
$$Business\ Cost(Pattern\ 3) = 5.56$$

We assume that there is always a 50 percent chance of finding the flight according to the customer inputs. Hence the Business Cost of Pattern 2 and Pattern 3 would then be:

$$Business\ Cost(Pattern\ 2\ with\ Pattern\ 3) = 61.9 * 0.5 + 5.56 * 0.5 = 33.71$$

Hence the total Business Cost of the BPD would then be:

$$Business\ Cost(Pattern\ 1\ (Pattern\ 2\ with\ Pattern\ 3)) = 20.1 + 33.71 = 53.8$$

Impact of Resequencing of Tasks. We see from the calculations that the change in the sequence leads to a change in the Business Cost. Nevertheless the rest of the parameters i.e Cost and Reliability of the process do not change.

3.2 Knock Out Order: 'Knock-Out in an Increasing Order of Effort and a Decreasing Order of Termination Probability

Every Business Process has conditions that need to be checked. If the conditions are not fulfilled then the execution of the process is terminated. This best practice recommends that conditions that have the highest probability to terminate the process should be executed right at the begining, followed by the condition having the next highest probability to terminate the process and continue. In such a case the reasons why a Business Process needs to be terminated is accomplished at the very begining and the rest of the Business Process is executed with a high probability of achieving the Business value. The knock-out best practice is a variant of the resequencing best practice. Van der Aalst [4], [5], [6], [7] mentions this best practice and also gives quantitative support for its optimality.

We believe that every task in the Business Process has a certain Reliability with which it performs. Hence the interpretation of this best practice in our case would mean that the Business Cost of the process will be lower in case the initial part of the process has a lower Reliability than the latter. To evaluate this we consider the Business Process to book a flight, nevertheless we will assume that all the flights are available. Hence there is no condition involved to check for the availability of the flight. This is shown in the Fig. 7.

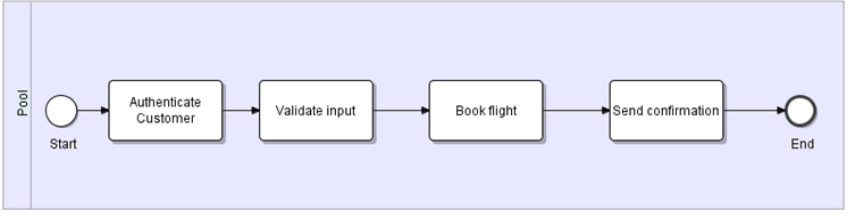


Fig. 7. Business Process Diagram to book a flight

Original Case - Task Order with Descending Reliability. We make assumptions on the Cost and Reliability such that the Reliability of the tasks have a descending order. We assume the Costs are the same on each of the tasks. The calculation of the Business Cost is as shown in the table 5.

Table 5. Flight Booking - Task Order with Descending Reliability

Task	Cost	Reliability	Business Cost of the task	Business Cost
Authenticate Customer	10	0.9	11.11	11 .11
Validate Input	10	0.8	12.50	26.4
Book Flight	10	0.7	14.29	52.0
Send Confirmation	10	0.6	16.67	103.3

Hence the Business Cost with descending order of Reliability would be:

$$Business\ Cost(Descending\ Reliability) = 103.3 \tag{13}$$

Changed Case - Task Order with Ascending Reliability. We make assumptions on the Cost and Reliability such that the Reliability of the tasks now have an ascending order. The calculation of the Business Cost is as shown in the table 6:

Table 6. Flight Booking - Task Order with Ascending Reliability

Task	Cost	Reliability	Business Cost of the task	Business Cost
Authenticate Customer	10	0.6	16.67	16.67
Validate Input	10	0.7	14.29	38.1
Book Flight	10	0.8	12.50	60.1
Send Confirmation	10	0.9	11.11	77.9

Hence the Business Cost with ascending order of Reliability would be:

$$Business\ Cost(Ascending\ Reliability) = 77.9 \tag{14}$$

Impact of Knock-Out Order: The example shows that the Knock-out Order brings the Business Cost of the Business Process down. The Knock-out sequence push the tasks which have the highest probability of terminating the process to the front and in turn makes sure that the rest of the process is executed only when all the conditions are met. The best practice does not change the overall Reliability and Cost of the process.

3.3 Task Elimination: Eliminate Unnecessary Tasks from a Business Process

This best practice recommends that the tasks which are having no value or tasks which are redundant should be eliminated. A task in a Business Process when eliminated reduces the Business Cost of the process. In case tasks in a Business Process are eliminated from the optimization perspective, this will then lead to a compromise on the quality of the process. There are different ways in which an evaluation can be done so as to find if tasks are unnecessary or redundant. One way is to look into tasks which consider iterations. Iterations indicate that a certain task is done "n" number of times because it has not achieved the Business value at once. Tasks redundancy can also be considered as a specific case of task elimination. In order to identify redundant tasks, Castano et al [8] have developed entity-based similarity coefficients.

We consider an example to book a hotel to evaluate this best practice. Consider a Business Process where an agency tries to find a room in a hotel according to the inputs given by the customer. Finding a room in a hotel is an iterative process. The travel agency nevertheless would like to try in every hotel possible

to find a room until a room is found. Hence this task would be executed iteratively until the Business objective is met. In case every loop in this iteration Costs some money, the travel agency will need to decide on the number of hotels that they are willing to contact to find a room. The Cost of a task in a Business Process which has a looping to provide for Business Reliability varies according to the order in which each of the providers is called for. We use the BPD from the hotel booking process (see Fig. 8) in this case. Let's assume that there are six hotels with which the process interacts in a sequential order.

To check for the variations it could bring in the Cost we make assumptions here such as: the hotel which provides the highest Reliability also has the highest service charges or Costs.

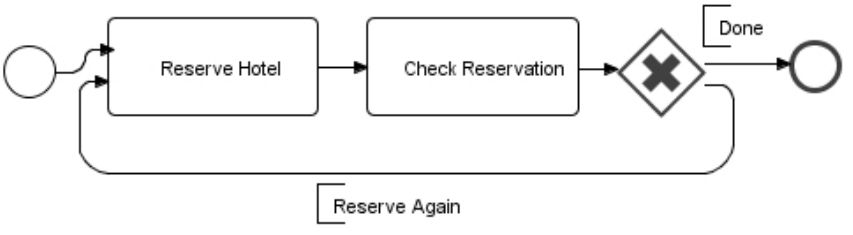


Fig. 8. Detailed BPD for booking a hotel

We execute this iteration in two scenarios:

Scenario 1: Highest Reliability - Highest Cost In this scenario we lay highest priority on the Reliability of the called service. Table 7 shows the hotels, there Reliability, Costs etc. due to the iterative condition every new hotel which is on the list increases the reliability. This increase in the reliability in case of n iterations is calculated as following:

$$Reliability(R1, Rn) \;=\; 1 \;-\; ((1 - R1) * (1 - R2)...(1 - Rn))$$

We call this reliability as the cumulative reliability and this is shown in the table.

Scenario 2: Achieve minimum increase in actual Cost with increase in Reliability In this scenario we start with the least reliable hotel and then select hotels with ascending order of Reliability. Table 8 shows the hotels with the development of Cost, Reliability, etc.

Impact of Task Elimination. From the calculation as shown in the table 7, for Scenario 1 adding the sixth hotel on the list increases the Reliability of the Business objective to find a room in the hotel from 0.998 to 0.999. Similarly in Scenario 2 the Reliability increases by very small percentage between the 5th and 6th hotel. Also, in scenario 1 we see that we check only the first hotel to reach a reliability of 0.9 whereas in the scenario 2 four hotels need to be checked

Table 7. Sample values

Option	Cost	Reliability	Cumulative-Rel	Actual Cost	Business Cost
Hotel 1	6	0.9	0.9000	6.00	6.67
Hotel 2	5	0.8	0.9800	6.500	6.633
Hotel 3	4	0.7	0.9940	6.580	6.620
Hotel 4	3	0.6	0.9976	6.598	6.614
Hotel 5	2	0.5	0.9988	6.603	6.611
Hotel 6	1	0.4	0.9992	6.604	6.609

Table 8. Sample values

Option	Cost	Reliability	Cumulative-Rel	Actual Cost	Business Cost
Hotel 6	1	0.4	0.4000	1.000	2.500
Hotel 5	2	0.5	0.7000	2.200	3.143
Hotel 4	3	0.6	0.8800	3.100	3,523
Hotel 3	4	0.7	0.9640	3580	3,714
Hotel 2	5	0.8	0.9928	3.760	3,787
Hotel 1	6	0.9	0.9992	3.803	3,806

before a reliability of 0.9 is achieved. Nevertheless, in both the cases the Business Cost does not increase by a huge margin and hence this could be an option to keep the iteration. But this situation could also be because the Cost for each of the iteration is coming down in comparison to the previous iteration. This leads to the situation where it might be that the last iteration need not be executed at all. In other words this redundancy can be eliminated and in turn there will be no or very less impact on the Business Process or the Costs that are involved.

3.4 Order Type and Triage

The best practice "Order type" says: determine whether tasks are related to the same type of order and, if necessary, distinguish new Business processes and the best practice "Triage" says consider the division of a general task into two or more alternative tasks or consider the integration of two or more alternative tasks into one general task. Both these best practices are similar to each other, at least in their intentions. Both of them are recommended so as to improve quality and in turn reduce Costs by either breaking tasks into many or by grouping certain tasks together.

Both these best practices are mentioned by a host of researchers which includes [9], [2], etc.

Original Case. To evaluate this Business Process we use the process for booking a flight which is shown in Fig. 7. The Business Cost of the process on a sample set of Cost and Reliability values is as shown in the table 6.

Changed Case. For the evaluation process we execute the tasks "Book Flight" and "Send Confirmation" together as one task assuming that the Cost of the new task is a summation of the Costs of both the tasks and the Reliability is the product of the reliabilities of the two tasks. In such a case the corresponding Business Cost is as shown in the table 9.

Table 9. Flight Booking

Task	Cost	Reliability	Bus.Cost of task	Business Cost
Authenticate Customer	10	0.6	16.67	16.67
Validate Input	10	0.7	14.29	38.1
Book Flight and Send Confirmation	20	0.72	27.78	80.7

Impact of Order type and Triage. We see that the Business Cost has increased when we put the tasks together. This is because the combined Reliability of the task is less than the two individual tasks. Combining two tasks might increase the quality and increase optimization, nevertheless this doesn't necessarily mean that the Cost of the process decreases. The combined task will produce a reduction in the Business Cost when:

$$Cost(New\ task)\ <=\ Cost(A)\ +\ Cost(B) \tag{15}$$

$$Rel(New\ task)\ >=\ Rel(A)\ *\ Rel(B) \tag{16}$$

3.5 Parallelism: Consider Whether Tasks May Be Executed in Parallel

This best practice recommends execution of tasks parallely rather than in sequential order. By doing this there is an effect on the Business Cost, probably bringing it down. At the same time the quality and co-ordination efforts increase due to the parallel execution of the tasks.

Original Case. In order to evaluate this best practice we take the Business Process to book a hotel and a flight depending upon customer inputs. We execute this process in sequential order to evaluate the impact on the Business Costs. The process is as shown in Fig. 9.

We make assumptions on Cost and Reliability. The table 10 shows the the calculation of the Business Cost on these assumptions.

From our calculations, the Business Cost of doing this process in a sequential order is 169.9.

Changed Case. Now we consider the same process in parallel order, we do the tasks "Book Hotel" and "Book Flight" in parallel to each other. This is shown in the Fig. 10. The pattern division is also as shown in the figure.

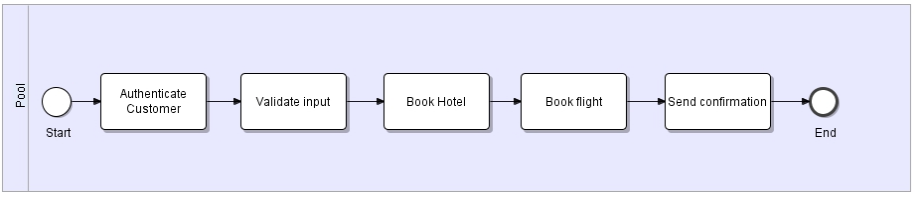


Fig. 9. Hotel and Flight Booking in Sequential Order

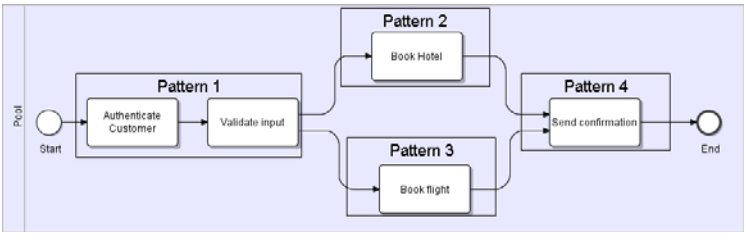


Fig. 10. Hotel and Flight Booking in Sequential Order

Table 10. Hotel and Flight Booking in Sequential Order

Task	Cost	Reliability	Business Cost of the task	Business Cost
Authenticate	10	0.9	11.11	11.11
Validate input	15	0.9	16.67	29
Book Hotel	20	0.7	28.57	70
Book Flight	30	0.7	42.86	142.9
Send confirmation	10	0.9	11.11	169.9

Table 11. Hotel and Flight Booking in Parallel Order - Pattern 1

Task	Cost	Reliability	Business Cost of the task	Business Cost
Authenticate	10	0.9	11.11	11.11
Validate input	15	0.9	16.67	29

Table 12. Hotel and Flight Booking in Parallel Order - Pattern 2

Task	Cost	Reliability	Business Cost of the task	Business Cost
Book Hotel	20	0.7	28.57	28.57

The tables 11 12 13 14 shows the calculation of the Business Cost according to the patterns.

Table 13. Hotel and Flight Booking in Parallel Order - Pattern 3

Task	Cost	Reliability	Business Cost of the task	Business Cost
Book Flight	30	0.7	42.86	42.86

Table 14. Hotel and Flight Booking in Parallel Order - Pattern 4

Task	Cost	Reliability	Business Cost of the task	Business Cost
Send confirmation	10	0.9	11.11	11.11

As Pattern 2 and Pattern 3 are in parallel, the Business Cost for both together is as shown in table 15:

Table 15. Hotel and Flight Booking in Parallel Order - Pattern 2 —— Pattern 3

Task	Cost	Reliability	Business Cost of the task	Business Cost
Pattern 2 —— 3	50	0.49	71.43	71.43

The total Business Cost by breaking the process in parallel is as shown in table 16:

Table 16. Hotel and Flight Booking in Parallel Order

Task	Cost	Reliability	Business Cost of the task	Business Cost
Pattern 1	20	0.81	24.69	24.69
Pattern 2 —— 3	50	0.49	102.04	152.4
Pattern 4	10	0.9	11.11	180.5

Impact of Parallelism. We see from the calculations that the Business Cost of the process increases by doing a process in parallel than by doing it in a sequential order. Execution of processes in parallel requires more tasks to make the process reach a logical end including compensation tasks. For example, in the Business Process that we considered, we will need a compensation task in case only the flight or only the hotel is booked. in this case the compensation will have to cancel the other booking. These situations do not arise when tasks are done in a sequential order.

3.6 Conclusion

In this paper we have evaluated the most commonly recommended best practices on Business Processes so as to determine their impact on Cost, Business Cost and Reliability before and after implementing the best practice. We see through this evaluation that these best practices achieve financial optimization, nevertheless the variation and the impact on the parameters cannot be generalized. These are

dependent on the process and the complexity that the process is handling. We also see that in certain cases the implementation of the best practice does not lead to any financial gains, instead it Costs more to control the process and keep the quality high. In this paper, we have shown these effects on simple Business cases. Nevertheless, to elaborate the effects of these best practices, we will have to consider realistic business cases which include conditions such as failure and compensation mechanisms.

References

1. Sampath, W.: Computing the Cost of Business Processes. In: Third International United Information Systems Conference, Sydney, Australia, vol. 12(2), p. 20 (2009)
2. Klein, M.: 10 principles of reengineering. Executive Excellence 12(2), 20 (1995)
3. Manganelli, R., Klein, M.: The reengineering handbook:a step-by-step guide to business transformation. American Management Association, New York (1994)
4. Van der Aalst, W.M.P., Berens, P.J.S.: Beyond workflow management:product-driven case handling. In: Ellis, S., et al. (eds.) International ACM SIGGROUP Conference on Supporting Group Work (GROUP 2001), pp. 42–51. ACM Press, New York (2001)
5. Van der Aalst, W.M.P.: Workflow verification: Finding control-flow errors using petri-net-based techniques. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, pp. 161–183. Springer, Heidelberg (2000)
6. Reijers, H.A., Limam, S., Van der Aalst, W.M.P.: Product-based workflow design. Journal of Management Information Systems 20(1), 229–262 (2003)
7. Van der Aalst, W.M.P.: Reengineering knock-out processes. Decision Support Systems 30(4), 451–468 (2000)
8. Castano, S., de Antonellis, V., Melchiori, M.: A methodology and tool environment for process analysis and reengineering. Data and Knowledge Engineering 31, 253–278 (1999)
9. Hammer, M., Champy, J.: Reengineering the Corporation: A Manifesto for Business Revolution. Nicholas Brealey Publishing, London (1993)
10. Rupp, R.O., Russell, J.R.: The golden rules of process redesign. Quality Progress 27(12), 85–92 (1994)
11. Peppard, J., Rowland, P.: The essence of business process reengineering. Prentice-Hall Editions, New York (1995)
12. Berg, A., Pottjewijd, P.: Workflow:continuous improvement by integral process management. Academic Service, Schoonhoven (1997) (in Dutch)
13. Clark, A., Gilmore, S.: Evaluating Quality of Service for Service Level Agreements (2007)
14. Fourneau, J.-M., Kloul, L.: A Precedence PEPA Model for Performance and Reliability Analysis
15. Bradley, J.T., Dingle, N., Gilmore, S.T., Knottenbelt, W.J.: Extracting Passage Time form PEPA models with the Hydra tool: a case study. In: UKPEW 2003 (2003)
16. Hillston, J., Kloul, L.: An efficient kronecker representation for PEPA models. In: de Luca, L., Gilmore, S. (eds.) PROBMIV 2001, PAPM-PROBMIV 2001, and PAPM 2001. LNCS, vol. 2165, p. 120. Springer, Heidelberg (2001)
17. Bravetti, M., Gilmore, S., Guidi, C., Tribastone, M.: Replicating Web Services for Scalability (2008)

Experience Driven Process Improvement

Mukhammad Andri Setiawan and Shazia Sadiq

School of Information Technology & Electrical Engineering,
The University of Queensland, Australia
{andri,shazia}@itee.uq.edu.au

Abstract. The importance of process improvement and the role that best practice reference models play in the achievement of process improvement are both well recognized. Best practice reference models are generally created by experts in the domain who are external to the organization. However, best practice can also be implicitly derived from the work practices of actual workers within the organisation, especially when there is opportunity for variance within the work, i.e. there may be different approaches to achieve the same process goal. In this paper, we propose to support process improvement intrinsically by utilizing the experiences and knowledge of business process users to inform and improve the current practices. The main challenge in this regard is identifying the "best" previous practices, which are often based on multiple criteria. To this end, we propose a method based on the skyline operator, which is applied on criteria relevant data derived from business process execution logs. We will demonstrate that the proposed method is capable to generate meaningful recommendations from large data sets in an efficient way, thereby effectively facilitating organizational learning and inherent process improvement.

Keywords: Process Improvement, Best Precedents, Flexible Processes, Multi Criteria Decision Making, Business Process Variants, Skyline Operator.

1 Background and Motivation

Process improvement continues to be the named number one priority for organisations [1]. Even though process improvement is typically solicited through expert advice and best practice reference models, a valuable and often overlooked source of best practice is the experiences and knowledge of individuals who perform various activities within the business process, and can be considered domain experts in a particular aspect of the overall operations. These experiences constitute the corporate skill base and should be considered a valuable information resource for organizational learning and process improvement.

Furthermore, business processes often face a dynamic environment which forces them to have the characteristic of ad-hocism in order to tailor to circumstances of individual process cases or instances. This creates business process variants [2], that is, the same process may have different approaches to achieve the same goals. The variants include the creativity and individualism of the knowledge worker, but are generally only tacitly available. Each variant has the same goal but by having

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 75–87, 2011.

different approaches, it may have different time needed, different task set and/or sequence and different cost, and consequently a different level of perceived success.

A traditional Business Process Management System is not generally capable to select best process precedents since all instances follow the same process model, and thus there is hardly any variance that can reflect individual/unique approaches. However, some complementary work can be found within the BPM community that long recognized the need to provide flexible business [3-5]. It is expected by having a flexible business process, an organisation can rapidly adjust their business process to suit the changes in the environment and thereby capitalize on opportunities and/or save on costs. But, having a flexible process is not always a solution to achieve the most efficient practice for the organisation. In fact, the more flexible the system, the more a (inexperienced) user may struggle to find the best approach to address a particular case. These users are required to have deep knowledge of the process they are working on if they are to be successful [6].

In this paper we will present an approach that is intended to assist such users and promote intrinsic process improvement. That is to facilitate change in practices by learning from already existing successful practices. The approach is intended to provide assistance to users that allows them to select the *best process* been done by previous (arguably experienced) users. Rather than forcing users to make design decisions to handle particular cases, we will use the existing knowledge within the organization to adopt practices that best meet the required criteria. Such an approach can guide the future user to improve productivity from both user perspective as well as organisational perspective.

The challenge in this regard is the identification of the so called best process variant from the potentially large record of instance executions. This identification is fundamentally dependent on the criteria that define *best.* These criteria are generally many and relate to different aspects of the process. These could include criteria such as cost (e.g. dollar value of a shipment process); time (e.g. time taken for an approval process); popularity (e.g the frequency of execution of a particular sequence of field tests in a complaints response process) and so on.

Characterizing the precedent process according to required criteria firstly requires extraction of the requisite data from execution logs. Fortunately, there is a significant body of knowledge on the extraction of data from execution logs, which includes process discovery [7], process similarity analysis [8-10] and general monitoring and analysis functions. In this paper, we rely on existing contributions in this regard, and focus instead on the problem of analysing multiple, contradicting criteria to identify the best process. Multi-criteria analysis is a known hard problem [11]. In particular for analysis across large populations of data (process instances), the efficiency of any approach used becomes exceedingly important. Accordingly we utilize a specific technique for multi-criteria analysis based on the skyline operator [12]. The skyline operator is conducive to the problem as it significantly reduces the search space through efficient identification of so called skyline points. The points are guaranteed to contain the best process and can subsequently be utilized by traditional multi-criteria analysis techniques to provide ranking and other results in a highly efficient way.

The remaining paper is organized as follows. In section 2 we present related work. In section 3, we introduce a general set of criterion for characterizing business

process. We present our approach for analysis of past process in section 4. We first present a general method for computing the skyline for past process, and subsequently use MCDM method to select the best past process. The results of the approach are presented and evaluated through an experimental data set. Finally in section 5 we provide a concluding discussion, limitations and future extensions of this work.

2 Related Work

The issue of managing business processes as an information resource was first raised in [13], which points out that process models should be regarded as intellectual assets of enterprises. Subsequently process discovery, analysis and diagnosis activities have been intensively studied. The key aim is to continuously improve the process and related practices, and collectively business process analysis (BPA) has been identified as an essential prerequisite for gradual and incremental organisational change [14]. In application domains where significant amount of variances are produced during business process execution, managing the resultant process variants and subsequently reusing the knowledge from the variants needs to be supported explicitly [2]. The source of the variant data is the system execution log that stores event-based data for traces of different process executions.

Various process mining techniques [7] have been proposed, aiming at process discovery, i.e. reconstructing meaningful process models from execution data. The reconstructed process models can then be used to facilitate a range of process redesign and auditing activities. One of the key contributions of process discovery research has been to provide methods and techniques to identify process similarity [8-10].

The work presented in this paper is set apart from the above in that the focus is on post discovery activities. Knowledge of as-is process models is an essential prerequisite. The subsequent re-design to achieve desired process performance goals is a highly challenging and interdisciplinary area of study. Benchmarking against industry best practices, and use of reference models are clearly widely utilized in this respect. However, generic models often have a weak fit with the organizational context and as such do not always provide an easy path to change for existing practices. In this paper, we advocate the use of organization internal knowledge for process improvement, such that the improvement is intrinsically driven and is by definition in synch with organizational context. There have been some contributions with a similar rationale. For example, [6] has developed a recommendation service as an add on to a current process mining application. It predicts the next step to be performed in a case by looking into the execution log. Similarly [15] propose to reuse activity patterns for subsequent process modelling.

A key and currently under-studied aspect of the problem is the definition of criteria that underpin the recommendation of the best process. In this paper, we will identify and define a set of criteria for characterization of the process, and subsequently use the criteria to efficiently analyse and rank the recommendation decision in a way that allows working communities to effectively utilize the results.

As there are several criteria that characterize processes, their analysis and ranking becomes a multi-criteria decision making (MCDM) problem. MCDM method is widely used for effective decision making, supporting decision makers to evaluate

and rank problems which incorporate multiple, and usually conflicting criteria [11]. MCDM acquires the best alternative from all possible alternatives with respect to given criteria.

Recently, Skyline computation has been introduced in multi criteria decision making applications [16, 17]. A skyline query has been devised to address the multi-criteria recommendation problem in large (relational) data sets to discover several good items among large number of candidates [18]. The concept is to filter out relatively poor candidates (tuples) which are dominated by some other candidates. Querying mechanisms for skyline computations have been intensively studied with several contributions targeting various forms of optimizations and efficiency gains. In this paper, we will utilize the skyline approach as it provides necessary support for large populations of process variants as can be expected in a typical BPM installation. Further details regarding research on Skyline queries are provided in Section 4.

3 Criteria for Selection

Analysing and/or monitoring a given process against criteria such as time or cost is widely available through business process analysis tools. However there can be a number of criteria that characterize the processes, and are in turn used for process improvement. In general, improving business process will usually have goals, such as reducing costs, improving productivity, improving competitiveness, and reducing service or production time [19]. There is a large body of knowledge on business process assessment and improvement strategies.

In an application wherein there is some degree of flexibility in execution, a process user will endeavour to learn from previous precedents, but may struggle to identify the most suitable or efficient precedent. Identification of the relevant decision criteria is thus fundamental to promote knowledge sharing and transfer. In addition, the criteria need to be measurable or quantifiable. Accordingly, in this paper we identify a set of general criteria consisting of *efficiency*, *cost*, *popularity*, and *currency*. The criteria are not exhaustive and may be extended (see concluding discussion), but are utilized in this paper to maintain a manageable scope and convey the workings of the proposed approach.

Below we present an explanation on each of the considered criteria and some basic definitions.

Definition 1 (Process Model). Process Model is a specific process representation (for a set of process instances). The model is either explicitly designed by previous process designers in a flexible business process management system [9] or mined from the execution logs. All process models are called as *variants* which have the same overall goal[1] i.e. they belong to the same domain such as a customer response, or an insurance claim process.

Definition 2 (Process Instance). From each defined or reconstructed process variant model, there can be a number of process instances captured within the execution log, assumed to contain execution information such as the set of tasks involved in the

[1] Models may be grouped based on behavioral similarity, such as that considered in [8-10].

process execution, the sequence of task execution, the resources utilized in executing tasks, the process-relevant data, execution duration of the process instance and constituent tasks.

Definition 3 (Criteria Specific Derived Data). The Criteria Specific Derived Data (*CSDD*) is a data set retrieved from process execution logs and contains the computed values against designated criteria (presented below) for each completed process instance recorded in the log. Detail working of the calculations from execution log is omitted as they can be trivially observed.

Popularity. The popularity of the process model is a criterion that shows how many times a particular instance of specific process model (variant) has been selected by user/used previously. Process matching on structural similarity, such as that given by [9] is used to identify the various (groups of) variant models discovered.

Weight. The efficiency (with respect to time) and cost (with respect to resources utilized) criteria is collectively calculated as weight.

Currency. We indicate the currency of an instance through its start time, with the assumption that when an instance is initiated is essentially the time at which the initial decision on how to tackle the particular case was made. Other interpretations of currency can be used without impact on the analysis approach presented below.

4 Skyline for Process Analysis

The Skyline operator was introduced almost a decade ago [12], and has been extensively studied ever since due to its wide application in many applications including multi-criteria decision making [16]. The skyline is basically a set of points (representative of tuples), called skyline points, which are not dominated by another point in a given set of d-dimensional data points. A point dominates another point if it is as good or better in all dimensions and better in at least one dimension [20].

Definition 4 (Skyline). A point $p = (p[1], p[2], \ldots, p[d])$ is said to dominate another point $q = (q[1], q[2], \ldots, q[d]$ iff on every dimension $p[i] \leq q[i]$ (for $1 \leq i \leq d$) and on at least one dimension $p[j] < q[j]$ denoted as $p \prec q$. The skyline is a set of points which are not dominated by any other point.

The skyline points are computed through the *skyline query*. Suppose we are working the skyline for two criteria, *weight* and *popularity*. A skyline point demands a return of minimum value on weight and maximum value in popularity, thus a tuple is dominating if it has a lower value of weight and higher value of popularity than any other tuple. The skyline query will compare these values across the data population. The skyline is shown as in Fig. 1.

Many different variations have been introduced for the skyline query, see e.g. [16, 21]. In this paper, we base all skyline computations on the SFS (Sort Filter Skyline) method [22]. This method essentially pre-sorts the data points according to their scores obtained by a monotone function *f*, such that if $f(p) < f(q)$ then it is guaranteed that $p \prec q$. In other words, the function corresponds to a topological sort with respect to the dominance criteria. The SFS method is generally considered as baseline in

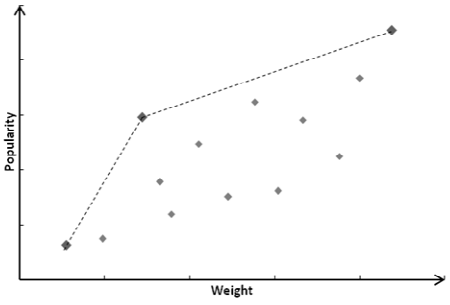


Fig. 1. Example of identifying skyline points from all candidates

benchmarking studies, hence it was suitable to demonstrate the implementation of the skyline query for this work.

In the subsequent sections, we will demonstrate how the skyline approach can assist in finding *best* process in an efficient way. The sections are presented against an example data set, which serves both as a motivating scenario as well as an experimental data set. We will first present the scenario and corresponding data set. We then present the skyline computation. In a large data set, the result of the skyline query may be a large number of points (instances). In order to further assist the user, we provide a method to rank the skyline result. Further, a user may not consider all criteria as equally important, e.g. cost may be more important than currency. We will demonstrate that in the ranking method utilized, user preferences on criteria are factored into the final ranking.

4.1 Scenario Description

We present an example of a business process in use at a real business to demonstrate the workings of our proposed method. The business process used is a bid tendering and completion process of a building services consultancy in Cairns, Australia. A building services consultancy usually deals with organisations looking to build new buildings or developments, e.g. a property developer, or organisations looking to retrofit existing buildings with new electrical, air-conditioning and communication infrastructure. For example local school buildings that need to be upgraded to cope with additional demand caused by increased enrolment and increased computer usage.

A *bid* represents a submission by the building services consultancy to an organisation looking for a contractor to undertake electrical and mechanical design work. This submission details what services will be rendered, and in what way.

First, the opportunity to submit a bid must be identified. This opportunity must be approved by management. If this is successful, the bid document must be drafted and submitted to the company that requested tenders. If the bid is successful, the work detailed in the bid must be completed. This work is then subjected to internal quality assurance mechanisms before it is released to the client. The final step in the process is to collect payment from the client.

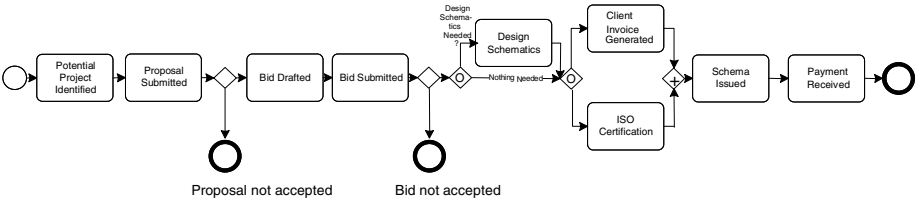


Fig. 2. Bid Tendering Process

On the bid tendering process above, process variants might exist e.g. whether design schema is needed or not (schema is needed if the similar works were never done before). The design schematics activities also can be broken down into few sub activities such as design general electrical schematics, design air-conditioning schematics, design fire protection relay circuit, and design acoustic schematics. The design schematics sub activities were recorded separately from the main execution log, and in the main execution log, all design schematics were recorded as one aggregated design schematics activity. There also exist parallel processes done while completing the process *Client Invoice Generated* and *ISO Certification*. Based on Fig. 2. above, an execution log of business process activities is collected. The result of the execution log shows some general data, overall process, and distributions of throughput time of each activity. Based on this initial data distribution on the real case study, a simulation tool was built to simulate the whole business process. The simulation tool was built to imitate the real case scenario, where the initial data distribution has shown some behaviour of how performers did the tasks e.g. some users completed a particular task faster than some other users; some activities were done in relatively the same amount of time spent. To test the scalability of the proposed system, we have generated execution log of business processes through our simulation tool which generates 10,000 process instances. From those 10,000 process instances, we collected 6,471 completed processes including variants that may exist in completing the process. Only completed process instances are considered to be the source of knowledge as they represent the information on how a process instance was done.

This record is partially shown in Table 1 below. The instance number i is represented as S_i. The *weight* property represents the time and cost value for the process instance. The *popularity* indicates the number of instances from the same/similar process variant model. Note that we use an additional attribute namely *Currency* which translates the *Date* into a numeric value representing a range. This is required as an instance created today does not mean it is 7 times better compared to a one week old instance, nor is it 365 times better compared to a one year old instance. Hence, we divide the date range of the oldest and the newest on (an arbitrarily chosen range of) one tenth basis and give 5 as the lowest value, continued with 5.1, 5.2, and so on until it reaches 5.9 (based on 1/10 increment). Clearly, the *Currency* computation can easily be tuned to suit the temporal properties of a given application e.g to make the granularity larger or finer depending on how sensitive the time interval needs to be made.

Table 1. Partial *CSDD* of Simulated Execution Data

Process Instance	Date	Currency	Weight	Popularity
S_1	27/10/2010	5	17.16	2273
S_2	28/10/2010	5	19.20	297
⋮	⋮	⋮	⋮	⋮
S_{4633}	25/11/2010	5.4	18.515	2273
⋮	⋮	⋮	⋮	⋮
S_{9997}	27/12/2010	5.9	16.79	2273
S_{10000}	27/12/2010	5.9	14.58	1480

We then implemented the *Sort First Skyline* method to be applied on the simulated data for the 3 criteria. The execution of the skyline query identifies 8 process instances as the skyline points which dominate all other points from 6472 completed processes, thus identifying the *best* tuples (instances) against the given criteria. These are given in Table 2.

Table 2. Instances on the Skyline

Process Instance	Currency	Weight	Popularity
S_{2387}	5.2	8.72309	1480
S_{6176}	5.6	8.98202	1480
S_{7939}	5.8	9.32156	1480
S_{8261}	5.8	8.27779	319
S_{9572}	5.9	9.70618	319
S_{9755}	5.9	10.1865	2273
S_{9939}	5.9	10.6003	1480
S_{9953}	5.9	10.7491	2273

4.2 Criteria Preferences and Ranking

In this section, we present a further refinement of the general method. As mentioned previously, in a large data set, the result of the skyline query may be a large number of points (instances). Thus user may not get a conclusive feedback regarding most relevant best instance as the recommended process. In order to further assist the user, we provide a method to rank the skyline result. Further, a user may not consider all criteria as equally important, e.g. cost may be more important than currency. Therefore, we include user preferences on criteria in the ranking procedure. Below we first provide the fundamentals behind the ranking procedure and then present its working using the above example.

One of the most widely used MCDM approaches is Simple Additive Weighting (SAW) [23], also called "*vector-maximum*" problem [24]. The concept is to obtain the

weighted summation of performance ratings on each alternative [25]. In most problems, SAW has shown an acceptable result and has a large following as it is easy to understand and implement [11]. The SAW method is based on the Multi-Attributive Decision Making (MADM) method, as defined below:

Definition 5 (Multi Attributive Decision Making). Generally the multi-attribute decision making model can be defined as follows [24]. Let $C = \{c_j \mid j = 1, \ldots, n\}$ the criterion set and let $A = \{a_i \mid i = 1, \ldots, m\}$ the selected set of alternatives (in our case process instances). A multi criteria decision making method will evaluate m alternatives A_i $(i=1,2,\ldots,m)$ against C_j $(j=1,2,\ldots n)$ where every criteria is independent of each other. A decision matrix, X, given as follows:

$$X = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \tag{1}$$

where x_{ij} is a rank of an alternative (process instance) i against criteria j. The preference weight is given as W which become the variables' coefficients for each criteria of interest, where $W = \{w_1, w_2, \ldots, w_n\}$. The preference weight is a value which represents the relative importance of each criterion. Most approaches in MADM involve two basic stages; (1) scale the values of all criteria to make them comparable (normalization); (2) rank ordering of the decision alternatives accordingly.

Definition 6 (Simple Additive Weighting Method). To allow comparisons across the attributes using the SAW method, all the decision matrix elements are first comparably scaled (or normalized) using equation 2.

$$r_{ij} = \begin{cases} \dfrac{x_{ij}}{Max\, x_{ij}} \text{ if } j \text{ is a benefit attribute} \\ \dfrac{\text{Min } x_{ij}}{x_{ij}} \text{ if } j \text{ is a cost attribute} \end{cases} ; i = 1, 2, \ldots, m; j = 1, 2, \ldots, n. \tag{2}$$

where r_{ij} $(0 < r_{ij} < 1)$ is normalized rating of selected (instance) alternative against attribute/criteria. Each selected instance will thus have a preference value V_i, where

$$V_i = \sum_{j=1}^{n} w_j r_{ij} \quad ; i = 1, 2, \ldots, m; j = 1, 2, \ldots, n \tag{3}$$

The preference value V_j indicates how to rank the (instances) alternatives. The higher the V_j value is, the more preferred the alternative is.

Ranking on the Skyline: Although the SAW method provides the ability to factor in user preferences as well as provide a ranking on the results, it utilizes pair-wise comparisons which can be rather inefficient in large data sets. The implementation of the

skyline query allows the reduction of the search space that is the numbers of tuples to be considered by SAW are reduced from 6472 to 8.

Even though some tuples among the skyline may not be of direct interest to the user, every best alternative with respect to the user's implicit preferences is guaranteed to be present. On applying the SAW method to the 8 tuples identified through the skyline query, with an arbitrary defined preferences weight $W = (6,3,2)$, the ranking results shown in the Table 3 are obtained.

Table 3. Ranking on Skyline

Instance	V_j
S_{9755}	9.84
S_{9953}	9.62
S_{2387}	9.38
S_{6176}	9.35
S_{7939}	9.21
S_{9939}	8.64
S_{8261}	8.35
S_{9572}	7.50

Using SAW implementation, we found that S_{9755} is the preferred alternative as it has the highest rank. This result is consistent with the result of SAW calculation on all recorded logs without skyline filtering.

The result of the ranking is then delivered as source of recommendation in the experience driven process improvement, which will let users learn the experience given by the best process on how the instance were done.

4.3 Method Evaluation

The sorting process in SFS takes $O(n \log n)$ time, and the skyline generation based on the sorted list takes $O(N \cdot n)$ time, where n is the number of tuples in the data set and N is the number of attributes [21]. However, as we detailed above, skylines may not entirely satisfy the user's expectations as it does not factor in user preferences and neither does it provide a ranking even though it able to select the best candidates. The MCDM/SAW processes can address the problem effectively. However, the SAW process will require pair-wise comparisons which for a large data set can be prohibitive. The approach presented above which combines the two therefore provides an efficient means of reaching the objective of identifying best process variants/instances.

Further there is some evidence that the wide application of skyline queries warrants its inclusion as a standard feature in commercial relational DBMS [26], which adds to the practical value of the proposed approach.

Our study is not without limitations. There can be circumstances wherein the skyline query may produce an empty result, e.g. in cyclic dominance relations. In cases

where data is characterized as above, the efficiency gains of the approach will not be available for the variant ranking computation. Furthermore the developed method only computes a single result, which might not fit all users e.g. new users might not be able to use the best past practice as the reference/guidance for their current practice as they have not gained yet the skills and the knowledge needed to perform the recommended best practice e.g. some complex activities need to be performed only by those who have the knowledge (In our real case world, in the bid tendering process mentioned previously, a design schematic activity is considered as a complex activity and should be avoided by new users). A gradual learning process from novice level to expert level might be more suitable for new users rather than to force them to follow a high standard defined by expert performer as shown implicitly in the best process precedent. This will let users to achieve better performance which is still on their current capabilities.

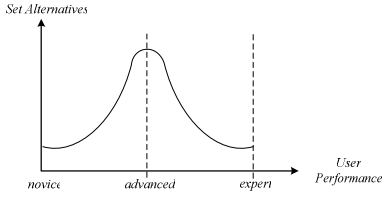


Fig. 3. Users' Current Performance level

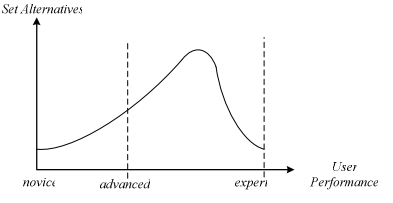


Fig. 4. User's Expected Future Performance level

Consider both graphs shown in Fig. 3 and Fig. 4. Graph in Fig. 3 shows an example of a scale of performance of set of alternatives (the business process instances) that are ranked and will work as the guidance for users to perform their activities. Based on standard distribution, only few users are considered as expert, and many users are finding them below the advanced level. A graph shown by Fig. 4 shows how in the future, overall performance level of users will be increased once users learn from the recommendation of the given best practices where we could find that more users are positioned well than the advanced level setup previously.

The role of process variants is inherent in the above, as it is the variance which represents the diversity in user practices. Detection of not only the best variant, but the best variant with respect to a particular users' current performance is a critical component of our approach.

We also note that a general skyline may not be entirely useful in a circumstance where some user seeking precedents against specific process characteristics. For example, a user may be specifically looking into a process where involving specific customers e.g. government client vs. private company client in the bid tendering process. A further relevance refinement might need to be done to accommodate this situation. This relevance of a instance relates to specific values of key *decision variables* that influence the *moment of choice* [27] in the process execution.

Lastly, we have considered four criteria in this work, and demonstrated the feasibility on a simulated data set of close to a thousand instances. It is conceivable that various applications characterize variant precedents by further or different criteria.

5 Conclusion

The availability of smart recommendation systems sensitive to user needs and behaviour especially on the world wide web, e.g. through meta search engines, has made the next generation user expectant of similar functionality from enterprise software as well. On the other hand, in spite of extensive induction procedures and training on best practices, there is evidence that (new) users may struggle to identify best course of action in areas of flexible practices. In this paper, we have endeavoured to match expectations and needs of the process user community by providing an effective and efficient means of capitalizing on previous practices and experiences within the organization. The proposed approach promotes intrinsic process improvement and change, leading to a *socialization of work practice* that is beneficial to both the individual user, as well as the organization.

Creating explicit recommendations from large data against multiple and conflicting criteria is a computationally hard problem. We have addressed this problem through a two-pronged approach that first reduces the problem space through the use of skyline queries and further generates user preference specific rankings through the SAW method, thereby achieving a holistic but efficient solution.

A most important aspect of this approach is the criteria used in the analysis. So far we identified four criteria based on a survey of literature, namely cost, efficiency, currency and popularity. In our future work, we plan to extend the criteria in the decision-making framework, particularly for relevance filtering as described above. Further experiments also needed in the future by using the extended criteria set, and larger populations of instances in order to identify and address any scalability issues in the proposed approach. We also plan to provide the recommendation to individual user in such a way that it fits the individual user's current level of experience.

References

1. Gartner: Meeting the Challenge: The 2009 CIO Agenda. Egham, UK (2009)
2. Lu, R., Sadiq, S.K.: Managing process variants as an information resource. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 426–431. Springer, Heidelberg (2006)
3. Reichert, M., Rinderle, S., Dadam, P.: ADEPT workflow management system: In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 370–379. Springer, Heidelberg (2003)
4. Aalst, W.M.P.v.d.: Flexible Workflow Management Systems: An Approach Based on Generic Process Models. In: Database and Expert Systems Applications, 818–818 (1999)
5. Sadiq, S., Sadiq, W., Orlowska, M.: A Framework for Constraint Specification and Validation in Flexible Workflows. Information Systems 30 (2005)
6. Schonenberg, H., Weber, B., van Dongen, B.F., van der Aalst, W.M.P.: Supporting flexible processes through recommendations based on history. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 51–66. Springer, Heidelberg (2008)
7. Aalst, W.M.P.v.d., Dongen, B.F.v., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow mining: a survey of issues and approaches. Data Knowl. Eng. 47, 237–267 (2003)

8. van der Aalst, W.M.P., de Medeiros, A.K.A., Weijters, A.J.M.M.T.: Process equivalence: Comparing two process models based on observed behavior. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 129–144. Springer, Heidelberg (2006)
9. Lu, R., Sadiq, S.K.: 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)
10. van Dongen, B., Dijkman, R., Mendling, J.: Measuring Similarity between Business Process Models. In: Advanced Information Systems Engineering, pp. 450–464 (2008)
11. Hwang, C., Yoon, K.: Multiple attribute decision making: methods and applications: a state-of-the-art survey. Springer, Heidelberg (1981)
12. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE (2001)
13. Leymann, F., Altenhuber, W.: Managing business processes as an information resource. IBM Syst. J. 33, 326–348 (1994)
14. Biazzo, S.: Approaches to business process analysis: a review. Business Process Management Journal 6, 99–112 (2000)
15. Thom, L., Reichert, M., Chiao, C., Iochpe, C., Hess, G.: Inventing Less, Reusing More, and Adding Intelligence to Business Process Modeling. In: Database and Expert Systems Applications, pp. 837–850 (2008)
16. Lin, X., Yuan, Y., Zhang, Q., Zhang, Y.: Selecting stars: The k most representative skyline operator. In: ICDE, pp. 86–95. Citeseer (2007)
17. Wong, R.C.-W., Pei, J., Fu, A.W.-C., Wang, K.: Online skyline analysis with dynamic preferences on nominal attributes. IEEE Transactions on Knowledge and Data Engineering 21, 35 (2009)
18. Hsin-Hsien, L., Wei-Guang, T.: Incorporating Multi-Criteria Ratings in Recommendation Systems. In: IEEE International Conference on Information Reuse and Integration, IRI 2007, pp. 273–278 (2007)
19. Mansar, S., Reijers, H., Ounnar, F.: Development of a decision-making strategy to improve the efficiency of BPR. Expert Systems with Applications 36, 3248–3262 (2009)
20. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: an online algorithm for skyline queries. In: Proceedings of the 28th International Conference on Very Large Data Bases, VLDB Endowment, Hong Kong, China (2002)
21. Wong, R.C.-W., Fu, A.W.-C., Pei, J., Ho, Y.S., Wong, T., Liu, Y.: Efficient skyline querying with variable user preferences on nominal attributes. In: Proc. VLDB Endow., vol. 1, pp. 1032–1043 (2008)
22. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with Presorting. In: International Conference on Data Engineering (ICDE), pp. 717–717 (2003)
23. Yeh, C.H.: A Problem-based Selection of Multi-attribute Decision-making Methods. International Transactions in Operational Research 9, 169–181 (2002)
24. Zimmermann, H.J.: Fuzzy set theory–and its applications. Kluwer Academic Pub., Dordrecht (2001)
25. Fishburn, P.C.: Additive utilities with incomplete product sets: application to priorities and assignments. Operations Research 15, 537–542 (1967)
26. Eder, H., Wei, F.: Evaluation of skyline algorithms in PostgreSQL. In: Proceedings of the 2009 International Database Engineering & Applications Symposium. ACM, Cetraro-Calabria (2009)
27. Rozinat, A., van der Aalst, W.M.P.: Decision mining in proM. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 420–425. Springer, Heidelberg (2006)

Deep Business Optimization: Making Business Process Optimization Theory Work in Practice

Florian Niedermann and Holger Schwarz

Institute of Parallel and Distributed Systems, University of Stuttgart,
Universitätsstraße 38, 70569 Stuttgart, Germany
{florian.niedermann,holger.schwarz}@ipvs.uni-stuttgart.de
http://www.ipvs.uni-stuttgart.de

Abstract. The success of most of today's businesses is tied to the efficiency and effectiveness of their core processes. This importance has been recognized in research, leading to a wealth of sophisticated process optimization and analysis techniques. Their use in practice is, however, often limited as both the selection and the application of the appropriate techniques are challenging tasks. Hence, many techniques are not considered causing potentially significant opportunities of improvement not to be implemented. This paper proposes an approach to addressing this challenge using our deep Business Optimization Platform. By integrating a catalogue of formalized optimization techniques with data analysis and integration capabilities, it assists analysts both with the selection and the application of the most fitting optimization techniques for their specific situation. The paper presents both the concepts underlying this platform as well as its prototypical implementation.

Keywords: Business Process Optimization, Optimization Techniques, Business Process Analytics, Data Mining, Tool Support.

1 Introduction

In this section, we first present the approach that is usually employed during *Business Process Optimization (BPO)*. Then, using the introduced approach, we discuss the challenges that are typically encountered when transferring *BPO* research results into practice. Finally, we show how the *deep Business Optimization Platform (dBOP)* can help to close the gap between *BPO* research and practice.

1.1 Business Process Optimization

In the past decade, businesses have moved from tweaking individual business functions towards optimizing entire business processes. Originally, this trend - then called Business Process Reengineering [7] - was triggered by the growing significance of Information Technology and the trend towards globalization. The increasing volatility of the economic environment and competition amongst businesses has further increased its significance over the past years. Hence, the ability

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 88–102, 2011.

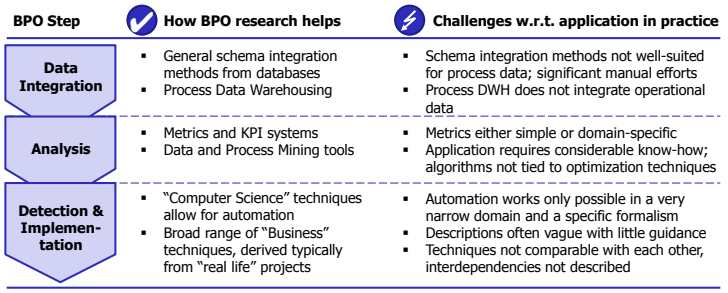


Fig. 1. Challenges for Applying BPO Research in Practice

of a business to achieve superior process performance is nowadays one of the main sources of competitive advantage (or, in many cases, survival of the business).

To achieve this, nearly all companies have dedicated *Business Process Optimization (BPO)* staff and frequently run large-scale optimization projects. Technically speaking, the ultimate goal of *BPO* is the selection of the right process designs and the application of the most appropriate optimization techniques. To achieve this goal, *BPO* efforts ideally include the following three steps:

1. **Data integration:** As processes are cross-functional, data pertaining to the process can be spread over many different data sources. Hence, as a first step, all possibly relevant data needs to be collected and integrated.
2. **Data analysis:** After the raw data has been collected, both the process model and the process data need to be analyzed. This analysis can range from the calculation of basic metrics (such as duration, cost or frequency) to the application of data mining techniques to discover "hidden" insights.
3. **Detection and implementation of improvements:** Based on the analysis results, deficiencies within the process are detected. These can, e.g., relate to the process flow, the composition of activities or the resources used during the process execution. After assessing the deficiencies, appropriate techniques for addressing are selected and applied to the process or its context (e.g., the resources executing the process).

1.2 BPO Challenges

Recognizing the importance of *BPO*, considerable research efforts both in Computer Science and Business have been made to address the challenges and complexities of each of the individual *BPO* steps listed in the previous section. While they have been successful in many ways of advancing the broad field of *BPO*, several factors limit their usability in a *BPO* project.

While the details of the limiting factors are shown in Fig. 1, they all can be traced to three common root causes. First, the data integration and analysis steps are often neglected. The optimization techniques assume that all relevant data is contained in a single data source. Further, typically only basic analysis techniques (such as metrics) are employed. Second, while there are plenty of

R1: Analytics	The approach needs to be able to integrate a broad range of heterogeneous data. Its analytics capabilities need to allow for the discovery of complex insights, even if the analyst is not highly familiar with data mining techniques.
R2: Optimization	To ensure that the most appropriate optimization techniques are used in any given scenario, the approach needs to be able to (semi-)automatically select and apply those techniques that match the description of the optimization goals given by the analyst. As it is likely that the analyst has additional knowledge about the process context, his input needs to be considered during the optimization.
R3: Integration	To facilitate adoption as well as minimize both the error rate and the need for human intervention, the various *BPO* steps need to be seamlessly integrated.

Fig. 2. *BPO* Usability Requirements

optimization techniques, each one is defined in isolation using a different formalism and/or with a different application domain in mind. This makes the combination of different techniques challenging, as their interdependencies are not defined and their results are sometimes not comparable. Finally, there is little to no integration between the different *BPO* steps, which makes especially the application of complex optimization techniques difficult.

1.3 The Deep Business Optimization Platform

To make *BPO* research work in practice, we need an approach that overcomes the challenges discussed in the previous section. Hence, we derive the requirements listed in Fig.2 for such an approach to be feasible.

Our *deep Business Optimization Platform (dBOP)* was developed with exactly these requirements in mind, which is why we use it throughout this paper as an example of how to achieve the paper's goal. The platform, as shown in Fig.3 consists of three integrated layers. The data integration layer handles the integration of heterogeneous data sources, with a specific focus on the integration of process and operational data. The integration layer is why we use the prefix "deep" as a qualifier for the platform name. It indicates that the optimization is based on an integrated, rather than an isolated view on the relevant data.

Based on the results of a graph analysis and knowledge about the employed optimization patterns, the analytics layer automatically processes and analyses the process data, both using standardized metrics and data mining techniques. Finally, the optimization layer utilizes an optimizer engine and the optimization patterns stored in the pattern catalogue to (semi-)automatically detect and apply process improvements.

Many of the *dBOP*'s technical details are extensively discussed in our previous work (see for instance [15], [12] and [13]). Hence, we the focus of this paper is to show how the different components work together to increase the usability of

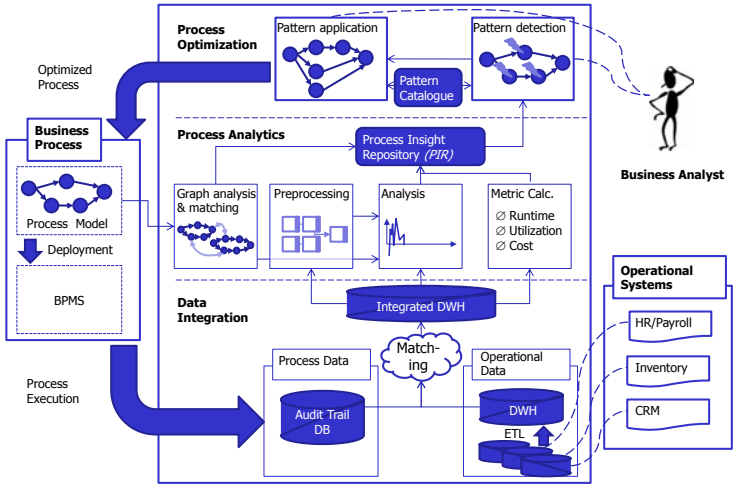


Fig. 3. dBOP Platform Overview

BPO research results by meeting the requirements laid out in Fig. 2. To that extent, this paper aims to give a balanced account of the platform's benefits from a user's and practitioner's perspective and the most salient concepts underlying it.

In Section 2, we discuss the integration and analytics capabilities and how they help with meeting requirement **R1**. Building on these results, Section 3 introduces both the process model and the pattern catalogue necessary for meeting requirement **R2**. In Section 4, we see how all the components come together in the prototypical implementation of the *dBOP* to fulfill requirement **R3**. After a brief evaluation and case study in Section 5, we discuss related work in Section 6 before concluding the paper in Section 7.

2 Data Integration and Analytics

As we have seen in the previous section, the first requirement for making *BPO* research work in practice is assisting the analyst with both the analysis of the process itself as well as the associated data. Therefore, we first take a look at how the *dBOP* uses data integration techniques to make sure that all relevant data is included in the analysis. Then, we present some of the data mining techniques employed and how they are matched to optimization techniques.

2.1 Data Integration

Broadly speaking, two different kinds of data are relevant for analyzing processes. To achieve integration, process activities pass data between each other. This *process data* is then typically stored in the audit trail database (see Fig. 3). This data is flow-oriented, i.e., it contains a complete picture of the dynamic

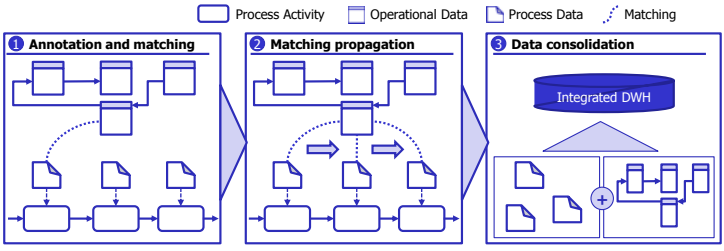


Fig. 4. Data Integration Approach

properties of the process (e.g., the process paths taken, the duration of the individual activities) but little information about the process subjects (such as the people and the items involved in it).

During their execution, processes additionally invoke one or several application systems which contain and generate data relevant to the process. This *operational data* is frequently consolidated in a Data Warehouse. It is subject-oriented, i.e., it contains a broad range of information about the process subjects, but little to no information about the process flow.

For a thorough process analysis, we need to integrate both kinds of data. During the design of the integration layer, we have found that two factors are imperative for this effort to succeed. First, the user needs to be assisted as much as possible with finding the most appropriate matches for integration. Second, the matching of the schema data needs to be combined with an according Data Warehouse structure and ETL (Extraction, Transformation, Load) [9] process.

Due to the different paradigms of process and operational data, classical schema matching approaches like [2] are not successful at meeting the requirements outlined above. This is why we have developed a specific approach for integrating operational and process data. As Fig. 4 illustrates, it consists of three steps: In the matching step, the analyst matches the attributes that he knows to be the same, optionally (if both the process and operational data is semantically annotated) supported by a semantic reasoner. These basic matches are then taken in the next step and spread to other process elements according to a set of processing rules. One of these rules, for instance, propagates the matchings along data flow mappings (such as the one realized by `<assign>` statements in BPEL). Finally, the data is consolidated into an integrated DWH, similar to the one discussed in [4] which is used as the main data source for analysis techniques [15].

2.2 Analytics

After the process and operational data have been successfully consolidated, it can now be analyzed together with the process model to discover insights that are useful for the optimization. The analysis is made up by four components, as can be seen in the analysis layer in Fig. 3. The first component analyses the process model and its execution behavior as well as matching it to other available process models. Together with the metrics calculation component, this indicates

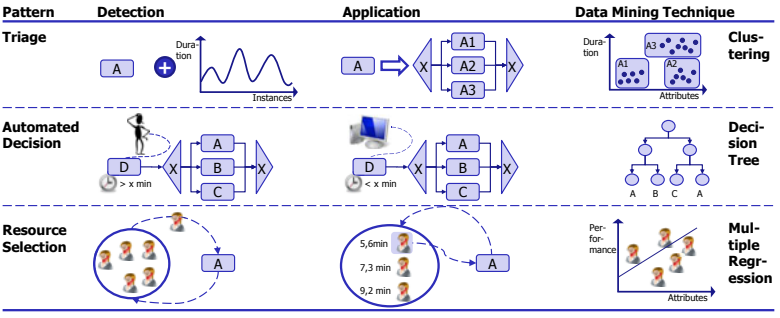


Fig. 5. Data Mining Techniques matched to Optimization Patterns

which areas of the process model might be interesting for further analysis. For these areas that have been selected as potentially interesting, the data is preprocessed and an in-depth analysis is performed using a suite of customized data mining algorithms.

As the spectrum of available data mining algorithms is huge [6], selecting the "right" algorithm is a challenging task that requires considerable analyst skills. To address this issue, each optimization pattern is internally matched to the suitable algorithm(s). This is exemplary shown in Fig.5 for three sample patterns.

The first pattern shown is the "Triage" pattern [16]. It is based on the notion that sometimes, during the execution of a certain activity, distinct variants of this activity emerge. In the interest of transparency and the ability to optimize the respective variants, the "Triage" pattern splits the activity. To determine if this pattern is applicable, the analyzer first checks if the activity's behavior (e.g., activity duration variance) suggests its presence. Then, a clustering algorithm, applied to different subsets of the activity's input data, tries to determine if there are "sufficiently different" [3] identifiable variants. If this is the case, the pattern can be applied.

In the middle of Fig.5 we see the "Automated Decision" pattern [11]. This pattern is based on the idea that in highly repetitive processes, a common pattern for manual decisions can be identified and the decision can hence be automated (or supported) by a classifier. For the application of this pattern, the analyzer first checks for all decisions in the process, if they exceed a certain duration (or cost, depending on the goal function). If that is the case, a decision tree (or alternatively, a multilayer perceptron [8]) for the decision is built and tested. If it is sufficiently good (as determined by the business analyst), it can be applied.

The third pattern displayed is the "Resource Selection" pattern. Its goal is to select, from a pool of resources, the one resource that execute the current activity "best" (e.g., fastest, with the highest quality, lowest cost - see Section 3.1), which is in contrast to the usual procedure, i.e., select any available resource randomly. For that purpose, a multiple (linear) regression model is used to predict the likely performance of any available resource. The one that shows the best likely performance for the given setup is selected (assuming that other defined constraints, such as utilization goals, are met).

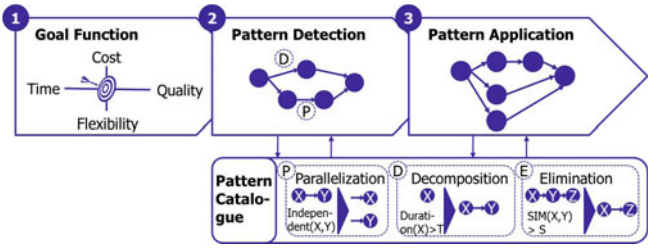


Fig. 6. *dBOP* Optimization Methodology

3 Optimization

In this section, we show how the *dBOP* uses the analysis results together with a set of formalized process optimization techniques for the semi-automated detection and implementation of improvement opportunities . First, we introduce the three-staged optimization approach and explain how it helps an analyst who is using it to achieve the desired results. Then, we present the criteria that are used to classify optimization patterns and give an excerpt of the pattern catalogue. Finally, we provide the detailed version of a sample pattern.

3.1 Optimization Overview

Each optimization pattern changes a process in a different way and with different effects. Further, many pattern have interdependencies that have an influence on, e.g., their order of execution. As there are currently more than 30 patterns to choose from, getting this right can be a daunting task for any analyst. This is why to make the application of these patterns practical (and to meet requirement **R2** of Fig.2), we have developed the three step methodology shown in Fig.6.

In the first step, the analyst is asked to select the goal function(s) he wants to achieve and define any constraints to apply to the optimization. In the next step, the optimizer tries to find instances of the patterns selected based on the previous step. This is achieved using a combination of metrics, graph analysis and data mining techniques, as we have seen in Section 2.2. In the final step, the analyst is presented with each pattern instance in an order that is likely to yield the optimal results. For those that he confirms, the process is modified accordingly.

3.2 Pattern Catalogue

At the core of the optimization methodology explained in the previous section is the pattern catalogue. In accordance with requirement **R2**, the pattern catalogue - an excerpt of which we see in Fig.7 - is a consistent collection of patterns which are mapped to a common formalism. To enable the optimizer to perform a qualified selection, the patterns are classified according to a wide range of criteria. First, each pattern is classified according to the type of changes it implements,

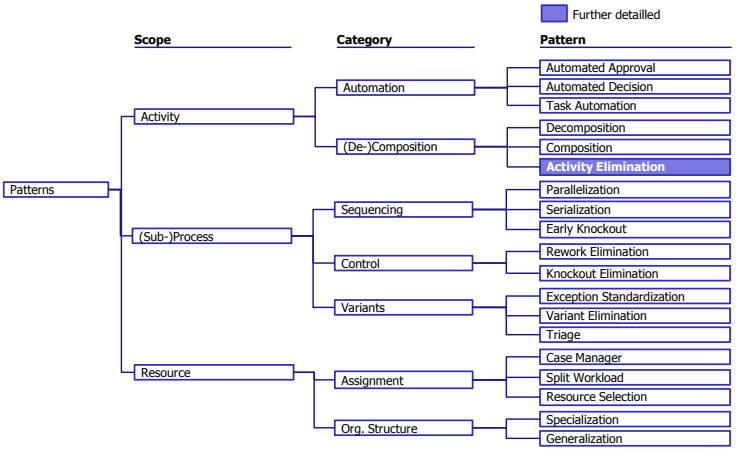


Fig. 7. Pattern Catalogue Excerpt

as shown in Fig. 7. Second, it lists whether a pattern contributes, prevents or is indifferent towards the fulfillment of each of the optimization goals. Third, it describes the process stage (design, execution, analysis) during which each pattern can be applied. Fourth, it lists constraints, especially w.r.t. the execution order that should be considered when combining this pattern with other patterns. Finally, it indicates which data (just the model, process or integrated data) is required to run the pattern.

More details on the classification of the patterns can be found in [14] and the next section, where we present a simplified version of the process meta-model as well as the description of a sample pattern.

3.3 Meta-model and Pattern Example

One of the key ingredients for defining the optimization patterns is a common process meta-model. In line with requirement **R2** our main goal with regards to the meta-model is to have a good mix of usability, proximity to common modeling languages such as BPMN and formal reasoning power. Hence, we have selected a graph based meta-model. The concepts presented can, of course, also be applied with minor modification to other meta-models like Petri nets.

The meta-model is based on the process model graph presented in [10]. In this paper we only introduce a greatly simplified version, so please see [10] and [14] for more details.

Definition. *Simplified PM Graph: A simplified PM Graph G is a tuple $(V, O, N, C, E, \iota, o)$ in which*

1. *V is the finite set of process data elements (also called variables).*
2. *O is the finite set of operational data relevant to the process.*

Pattern 1. Activity Elimination

Specification

Goals:	Time	↓	Cost	↓	Quality	→	Flexibility	→
Stage:	Design	✓	Execution	✗	Analysis	✓		
Data req.:	Model	✓	Process	(✓)	Operational	(✓)		

Detection

Require: Similarity threshold $T \in [0, 1]$, Activity nodes N_A of process graph G
Ensure: All found pattern *instances*

```
instances = {}
for all act ∈ N_A do
  for all succ ∈ Successors(act, E_C) do
    if SIM(act, succ) ≥ T then
      instances = instances ∪ newInstance(act, succ, SIM(act, succ))
    end if
  end for
end for
return instances
```

Application

Require: Graph G, all *instances*, selected *instance*, *analyst* input
Ensure: The optimized (improved) process graph G_{opt}

```
G_opt = G
if confirms(instance, analyst) then
  DeleteNode(inst.succ)
  RemoveDependents(instances, inst)
end if
return G_opt
```

3. *N is the finite set of process nodes which includes the set of activities N_A, the start and the end node, the termination nodes N_T and the control nodes (XOR and AND Fork/Join) N_C.*
4. *C is the finite set of conditions.*
5. *$E \subseteq N \cup N \cup C$ is the set of (control) connectors.*
6. *$\iota : N \cup C \cup \{G\} \rightarrow \wp(V)$ is the input data map, with $\wp(V)$ being the power set over V.*
7. *$o : N \cup \{G\} \rightarrow \wp(V)$ is the output data map.*

Using this meta-model, we can now define basic optimization patterns, such as the "Activity Elimination" pattern shown below in Pattern 1. As we can tell from the pattern specification, the elimination of redundant activities both lowers the process duration and cost. The pattern can be applied during the design and the analysis stage. It is flexible w.r.t. the employed data - the process model is sufficient, but if process and operational data is available, it can be used to further improve the required similarity measurement.

The idea of this pattern is that a high similarity of process activities can be an indicator for redundancy. To find similar activities, the pattern utilizes a

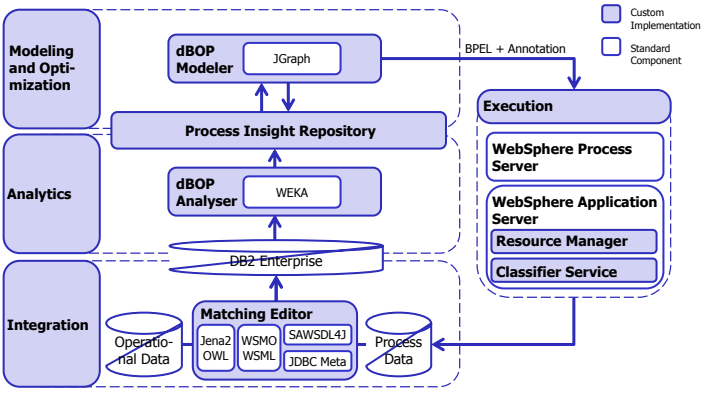


Fig. 8. dBOP Implementation Overview

similarity function $sim_A : N_A \times N_A \rightarrow [0, 1]$ [13], which measures the similarity of two activities on a scale of 0 (not similar at all) to 1 (identical) - in our experience, a threshold of 0.8 has shown a good correlation with human judgment. For those activities that are found highly similar, the analyst is asked to confirm whether they are redundant. If that is the case, the redundant activity is removed from the process.

4 Implementation and Prototype

After the previous sections have focused on explaining the concepts underlying the *dBOP*, we now take a closer look at its implementation. First, we give a brief overview over the different component implementations. Then, we present the *dBOP Modeler* in detail, as it is the core.

4.1 Implementation Overview

The implementation of the *dBOP* shown in Fig.8 achieves to fulfill the integration requirement **R3** by ensuring that the interfaces between the different components fit seamlessly into each other. Hence, while it is possible to use each component individually, transferring results is possible with no or minimal manual intervention.

The applications stack itself employs a mix of standard software and custom-developed applications

The *dBOP* **execution environment** is built on top of IBM WebSphere. The process execution itself occurs in IBM WebSphere Process Server, while run-time services such as the "Classifier" service for decision automation or the "Resource Manager" for resource allocation are built as enterprise applications and run on WebSphere Application Server. Note that instead of Process Server, we can run the process on any other BPEL (or for that matter YAWL-)compliant execution engine by providing a custom implementation of the mapping interfaces.

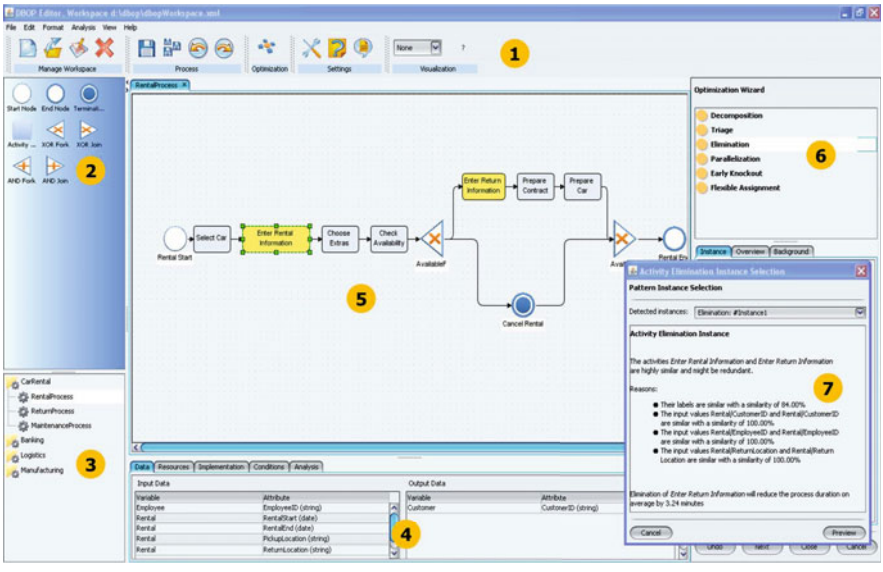


Fig. 9. *dBOP Modeler* Screenshot

The main component of the **integration layer** is the *Matching Editor* [15] which implements the matching approach described in section 2.1. For storing the integrated process and operational data, basically any sufficiently powerful relational database is feasible - our prototype uses DB2 Enterprise.

At the heart of the **analytics layer** is the *dBOP Analyzer*. It incorporates data preprocessing, metric calculation and data mining techniques to provide the necessary inputs for the optimization layer. Its implementation is based on a customized and extended version of the WEKA library [6].

The central component for the **modeling and the optimization** of the processes is the *dBOP Modeler* that we take a closer look at in the next section. For graph visualization and basic reasoning purposes, it uses the JGraph library [1] - beyond that, it is a custom implementation.

4.2 dBOP Modeler

After the overview of the whole platform's implementation in the previous section, we now take a closer look at one of the components - the *dBOP Modeler*. It is the central tool for process analysts, as it is used both for the modeling and the optimization of business processes. To allow for intuitive usage and a quick progression on the learning curve, the user interface, as shown in the screenshot in Fig. 9, has been designed to be as close to standard modeling tools as possible.

The basic features and options (1) of the *dBOP Modeler* closely resemble those of standard modeling tools. Further, it allows the analyst to visualize analysis results and explore the effects of different process setups. New elements can be inserted into the model by dragging them from the element palette (2). Process

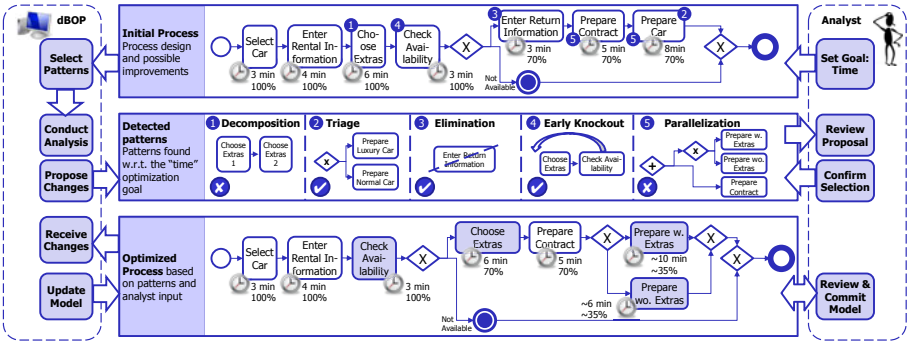


Fig. 10. Car Rental Case Study

models can be stored and loaded from the process repository (3), the contents of which can also be used for process model matching [13]. For enhancing the process model with data-, resource- and implementation information, additional facilities are provided (4).

The processes themselves are displayed in the modeling pane (5). As the *dBOP Modeler* is currently running in optimization mode, the modeling pane is visualizing the currently detected "Activity Elimination" pattern instance. This pattern was detected by the optimization wizard shown in (6) based on the optimization goal "time". It guides the analyst step-by-step through the optimization approach discussed above. Its current state shows for which patterns instances were found. Finally, before applying a pattern, the analyst gets detailed information about the expected effects (7) and, if applicable, possible conflicts that might arise from the application (e.g., when the "Activity Elimination" pattern would delete an activity whose outputs are used by a later activity).

5 Case Study and Evaluation

In this section, we briefly evaluate the *dBOP* in the light of the *BPO* requirements introduced in Fig.2. As the basis for our evaluation, we use the small "Car Rental" case study depicted in Fig.10 - the same process that can be seen in the screenshot in Fig.9.

In the case study, the analyst wants to optimize the process time. The *dBOP* hence analyses the process and the integrated DWH to find any patterns that either reduce process time or enhance general process attributes like transparency and manageability.

To improve transparency, the "Decomposition" pattern is proposed to split up the long running *Select Extras* activity. Further, a clustering of the *Prepare Car* execution results indicates that there are two distinct variants - one for cars with and one for cars without extras. To reduce the process duration, it is proposed to eliminate the possibly redundant "Enter Return Information" activity, move the knockout condition w.r.t the car availability [18] to the earliest possible position

and parallelize the "Prepare Contract" and "Prepare Car" activities, as the one is done by a mechanic and the other is done the rental clerk.

In this case, the analyst confirms all but the "Decomposition" and "Parallelization" patterns, reducing both the processing time by on average 3.9 minutes and improving the process transparency through the "Triage" pattern. The "Decomposition" pattern is not applied as the analyst does not see a possibility of sensibly splitting it. The "Parallelization" pattern remains unused as sometimes, the customer sometimes cancels the process during the contract phase and hence does not need the prepared car.

The case study illustrates, that the *dBOP* is a suitable approach for fulfilling the requirements of Fig. 2 and hence also suitable for transferring *BPO* research results into practice:

R1: Analytics	The *dBOP* automates most analytics and data integration steps. The analyst can hence focus on the reviewing the analysis results and the process optimization itself.
R2: Optimization	The patterns selection reflects the goal(s) and constraints defined by the user. The optimizer considers interdependencies between patterns. The analyst's superior knowledge of the process context is taken into account by leaving the decision about which pattern is applied with him.
R3: Integration	The *dBOP* ensures that all components work seamlessly together and that all data and instructions flow as require. Hence, the need for manual intervention is reduced only to those steps that can't be automated (such as giving "sensible" names to activities created by a split).

6 Related Work

As the *dBOP* main's goal is to improve the accessibility of existing research findings through integration, standardization and systematization, it is sensible to distinguish in the discussion of related work between the platform as a whole and the separate platform layers.

Looking at the *dBOP* as a whole, it can be seen as an application of cybernetics [19] to *BPO*. The workflow controlling framework discussed in [21] and the process analysis approach of [5] are somewhat similar to our platform in that they use custom analysis tools to gain process insights. However, their data integration capabilities are limited and they lack an integrated optimization layer.

Considering the integration of heterogeneous data sources as required for the integration layer, this is the classical domain of schema matching approaches such as [2] which, however, struggle with the specifics of process data. The integrated DWH is similar to the one presented in [4], however, with a stronger focus on including both process and operational data.

The analytics layer can be seen as a specific implementation of Business Process Analytics [20]. As this relatively new field has so far not yielded too

many readily applicable algorithms, we however by and large rely on "classical" data mining and preprocessing techniques [8]. Implementation-wise, the WEKA framework [6] is the core component used for fulfilling this layer's tasks.

The techniques used in the optimization layer rely heavily on existing *BPO* literature such as [7] and [17]. Particular important sources for our work are existing surveys on *BPO* techniques such as [16] and research into particular optimization techniques like [18]. We have used these papers as both a source of additional patterns and leveraged some of their findings, e.g., on the different optimization goals. Please note that the mentioned sources are only representative and by no means comprehensive, as the literature survey considered more than hundred different computer science, manufacturing engineering and business publications on the subject.

7 Conclusion and Future Work

This paper has argued that for making techniques derived from *BPO* research findings more readily applicable in practice, an approach is needed that assists analysts both in the selection and the application of the proper techniques. For this purpose, we have introduced our *deep Business Optimization Platform (dBOP)* that implements such an approach by mapping optimization techniques to a common formalism and combines them with specialized data integration and analytics capabilities. We have demonstrated how the *dBOP* hence empowers even analysts with only basic knowledge to apply sophisticated process analysis and optimization techniques.

As our prototypical implementation of the *dBOP* platform is by and large complete, our current focus is on showing its usefulness in "real world" application scenarios - both for providing a realistic assessment of its strenghts and a perspective on its limitations. For that purpose, we are using two different evaluation modes. For showing its benefits from a *BPO* effectiveness perspective, we are collaborating with several service and manufacturing companies on the application of the *dBOP* to their processes. To demonstrate its benefits with regards to analyst productivity, we are currently conducting a user study where we analyze the effect of the various *dBOP* components on the outcome of the optimization of selected sample scenarios.

References

1. Alder, G.: Design and implementation of the JGraph swing component. Techical Report, (6) (2003)
2. Aumüller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and ontology matching with COMA++. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (2005)
3. Berkhin, P.: A survey of clustering data mining techniques. Grouping Multidimensional Data, 25–71 (2006)

4. Casati, F., Castellanos, M., Dayal, U., Salazar, N.: A generic solution for warehousing business process data. In: Proceedings of the 33rd International Conference on Very Large Data Bases, pp. 1128–1137 (2007)
5. Castellanos, M., Casati, F., Dayal, U., Shan, M.C.: A comprehensive and automated approach to intelligent business processes execution analysis. Distributed and Parallel Databases 16(3), 239–273 (2004)
6. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: An update. ACM SIGKDD Explorations Newsletter 11(1), 10–18 (2009)
7. Hammer, M., Champy, J.: Reengineering the corporation: a manifesto for business revolution. Brealey, London (1993)
8. Han, J., Kamber, M.: Data mining: concepts and techniques. Morgan Kaufmann, San Francisco (2006)
9. Inmon, W.H.: Building the data warehouse. Wiley, Chichester (2009)
10. Leyman, F., Roller, D.: Production Workflow. Prentice-Hall, Englewood Cliffs (2000)
11. Niedermann, F., Maier, B., Radeschütz, S., Schwarz, H., Mitschang, B.: Automated process decision making based on integrated source data. In: Proceedings BIS 2011 (2011)
12. Niedermann, F., Radeschütz, S., Mitschang, B.: Deep business optimization: A platform for automated process optimization. In: Proceedings of the 3rd International Conference on Business Process and Services Computing (2010)
13. Niedermann, F., Radeschütz, S., Mitschang, B.: Design-time process optimization through optimization patterns and process model matching. In: Proceedings of the 12th IEEE Conference on Commerce and Enterprise Computing (2010)
14. Niedermann, F., Radeschütz, S., Mitschang, B.: Business process optimization using formalized patterns. In: Proceedings BIS 2011 (2011)
15. Radeschütz, S., Niedermann, F., Bischoff, W.: Biaeditor - matching process and operational data for a business impact analysis. In: Proceedings EDBT (2010)
16. Reijers, H.A., Mansar, S.L.: Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. Omega 33(4), 283–306 (2005)
17. Sharp, A., McDermott, P.: Workflow Modeling: Tools for Process Improvement and Applications Development. Artech House Publishers, Boston (2008)
18. Van der Aalst, W.M.P.: Re-engineering knock-out processes. Decision Support Systems 30(4), 451–468 (2001)
19. Wiener, N.: Cybernetics: Control and communication in the animal and the machine. MIT Press, Cambridge (1948)
20. zur Mühlen, M., Shapiro, R.: Business process analytics. Handbook on Business Process Management 2 (2009)
21. zur Mühlen, M.: Workflow-based process controlling: foundation, design, and application of workflow-driven process information systems. Logos Verlag (2004)

Flexible Artifact-Driven Automation of Product Design Processes

Ole Eckermann and Matthias Weidlich

Hasso Plattner Institute at the University of Potsdam, Germany
Ole.Eckermann@student.hpi.uni-potsdam.de,
Matthias.Weidlich@hpi.uni-potsdam.de

Abstract. Automated support of business processes by information systems can be seen as state-of-the-art for many domains, such as production planning or customer relationship management. A myriad of approaches to the automation of business processes in these domains has been proposed. However, these approaches are not suited for highly creative processes, as they are observed in the field of innovative product design. These processes require a high degree of flexibility of the process implementation. In this paper, we focus on product design processes and propose a methodology for the implementation of supporting workflows. In order to cope with the imposed flexibility requirements, we follow an artifact-centric approach. Based on high-level process models, object life-cycle models are derived. Those are manually enriched and used for automatic generation of an executable workflow model. We also present an implementation of our approach.

Keywords: process automation, flexibility, artifact-centric, object life-cycles, methodology.

1 Introduction

Since process orientation has been brought forward as a paradigm for structuring enterprises, process awareness emerged not only as an organizational principle, but had an impact on the design of information systems, cf., [1,2]. Process-aware information systems and workflow technology gained importance for the support of business processes in various domains [3], such as production planning or customer relationship management. This trend was manifested in a myriad of approaches to the automation of business processes and various standardization efforts as, for instance, WS-BPEL [4] and the WfMC reference model [5]. However, business processes in certain domains turn out to impose requirements on their implementation that are hard to address using common imperative workflow technology. An example for such a setting are clinical pathways that describe the different steps during interdisciplinary diagnosis and clinical treatment of patients in a hospital [6]. Information systems supporting this kind of processes, for instance, should allow for ad-hoc deviations of workflow instances. As such a feature is not supported by common workflow technology, specialized systems like the ADEPT system [7] have been developed.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 103–117, 2011.

In this paper, we focus on another example for processes that require a high-degree of flexibility of their implementation. That is, we consider the workflow support for highly creative processes as observed in the field of innovative product design. Although the actual design of consumer products is a manual task, the internal treatment of design proposals follows on predefined processes. Further, a high degree of repetition of these processes along with the need to track design decisions suggests to support such processes with according workflow technology. Unfortunately, processes for the treatment of product design proposals impose particular requirements on the underlying implementation, among them state-driven execution of activities and the possibility to react to externally triggered state changes of the proposals accordingly. These requirements suggest to approach the implementation of product design processes with the case-handling paradigm [8,9] or an artifact-driven approach [10,11]. We take up these ideas and show how they are adapted and extended for the concrete use case of product design processes.

Our contribution is a methodology for flexible artifact-driven automation of product design processes. With respect to the different involved stakeholders in product design processes, we propose two modeling perspectives to provide easy understandable process descriptions on the one hand and detailed technical specifications on the other hand. As design processes are artifact-centric we suggest object life-cycles for the technical specification of artifacts, while the high-level description is defined in a process modeling notation. Furthermore, we outline how both perspectives can be interrelated to apply consistency verification. While we rely on existing formalisms for the description of object life-cycles, we elaborate on object life-cycle composition and inheritance in detail. Finally, our approach comprises the generation of an executable workflow model with a structure that addresses the special needs of product design processes at runtime. Hence, our methodology covers all steps from the initial design of the overall processing to the specification of the supporting workflow. As an evaluation, we present an implementation of our approach.

The remainder of this paper is structured as follows. Section 2 outlines the characteristics of product design processes and defines requirements for process automation. We describe the levels of our methodology in Section 3 and elaborate on relations between models in Section 4. Our implementation is presented in Section 5. Finally, we review related work in Section 6 and conclude in Section 7.

2 Processes for Innovative Product-Design

Innovation is one of the key values for the success of almost every enterprise. Major potential for innovation lies in the process of designing new products. To get a maximum amount of promising product proposals, as many design processes as possible are performed. Thus, product design is a costly process that must be supported accordingly by information systems. Dealing with these processes involves some major challenges.

Artifact-centric process. A design process is characterized by intensive interdisciplinary collaboration, which requires artifact-centric views on running process instances. Various experts must be able to access the whole data available for certain proposals to be able to make decisions.

Many objects. To design a product, many ideas have to be evaluated and compared. Rather bad ones are rejected and resources are concentrated on promising ones. Furthermore, a product may be a composition of many different parts that are developed by different experts, but still influence each other.

State-driven enactment. Product design is a very creative process and must not be restricted by the underlying information system. Only the current state of a proposal determines the next activities, as it is very likely that certain activities have to be performed multiple times until their outcome is a suitable result. Besides, domain experts must be able to directly change properties of proposals, which might trigger state changes that cannot be anticipated.

Flexibility. Even if a proposal was accepted, it is still in progress. Certain tasks have to be performed again to incorporate new ideas and optimize the overall result. Therefore, redoing and skipping of tasks must be under the control of domain experts.

Impact of process environment. Product design processes may have long durations and take place in a constantly changing business environment. That makes it important to adapt processes fast. Additionally, changes might influence the goals of design processes and, therefore, should have direct impact on the execution of running process instances.

As the design process is driven by states of proposals, the actual process execution should be specified by object descriptions. Nevertheless, process models are helpful for the communication between stakeholders and for the specification of how business goals should be achieved, e.g. which design methods should be applied. Thus, a methodology for the automation of design processes should involve process specifications and detailed object descriptions. To maintain these models, which cover different perspectives and abstraction levels, methods for consistency verification must be provided. Design processes may become very complex as they comprise many different artifacts. A common approach to deal with such complexity is to create various models describing different artifacts and their behavior in different contexts. Consequently, there is the need for model composition to be able to derive one complete process specification. Finally, process execution must be state-based and flexible. Unforeseen external events must have direct impact on running process instances.

3 Methodology Levels for Artifact-Driven Process Automation

This section presents our methodology for the automation of product design processes. The three different levels are illustrated in Fig. 1. At the highest level,

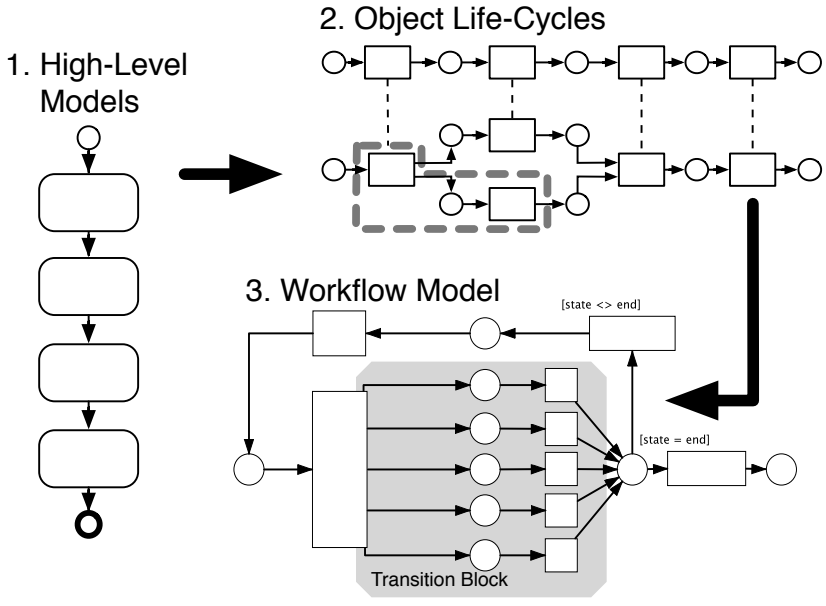


Fig. 1. Methodology for the automation of product design processes

business experts create high-level models (HLMs) to define how business goals of the company have be achieved w.r.t. the design of new products. At the second level, domain and IT experts model OLCs for all business entities involved in the process. In addition, they create relations between HLMs and OLCs and among OLCs, that describe similar behavior, specialization of behavior and dependencies between objects. Based on such relations we provide consistency verification for appropriate handling of frequently changing processes. The third level describes an executable workflow model that can be automatically derived from an OLC. The workflow model has a structure that enables purely state-based process execution and rich flexibility at runtime.

3.1 High-Level Models

Notations for business process modeling such as BPMN, EPCs, or UML activity diagrams have become state of the art for business process discovery and design [12,13]. They provide rich tools for the definition of an ordering of activities to achieve business goals on the one hand and role models to define responsibilities for execution on the other hand. Even though product design processes are data-driven, process models are very helpful. May it be discovery of processes, process documentation, specification of responsibilities or adaptation to and verification of compliance rules, it turned out that process models are the most suitable language for the communication between all involved stakeholders [2]. Therefore, we propose to start modeling product design processes with a common process description language to get an overview of the general procedure.

For illustration purposes, we use a BPMN subset containing activities, control nodes and edges to define such high-level models (HLMs). The restriction to a subset is necessary to be able to reason on the relation between HLMs and OLCs. For this subset, execution semantics are defined unambiguously by a translation into Petri nets models, cf., [14].

3.2 Object Life-Cycles

OLCs describe all states and state transitions of an object during its life time. States correspond to certain property sets but abstract from concrete data values, while state transitions represent business activities that have impact on the described object. State chart based notions like UML state machines and notions based on Petri nets are the most popular notions to model OLCs. The main difference between both notions is, that in a state machine an object state is represented by a node and, following on existing work [15,16], in a Petri net an object state is given by the *marking* of the net. In design processes, a product likely consists of various objects, which are developed in parallel. This has major impact on the respective OLCs, as it is necessary to define dependencies between those objects for a correct and complete system specification. Therefore, we us Petri net to define OLCs.

We interpret transitions as business activities and markings as states of an object handled by these activities. Accordingly, the Petri net must be safe as there is no reasonable interpretation in terms of an object state for a place of the Petri net that is marked with more than one token. Furthermore, we adopt the definition of OLCs introduced in [16]. An OLC has exactly one initial place corresponding to object creation and exactly one final place corresponding to object destruction. Each transition and each place is located on a path between these dedicated places.

OLCs describe the behavior of single object types. We assume that decisions at places, where alternative continuation is possible, are made based on the whole object state. This includes decisions made earlier in concurrent paths. We assume that potential deadlocks are avoided by the use of this information. Therefore, we consider OLCs to be correct, if they are relaxed sound [17]. That is, every transition has to be part of some firing sequence from the marking containing the initial place only to the marking containing the final place only.

3.3 Workflow Model

The third level of the methodology is a workflow model that enables rich flexibility during runtime and eases adaptation. The main idea behind the workflow model is to separate activities from the restrictiveness of control-flow and to determine execution orders purely state-based. That is, process execution is in general determined by the given OLCs and the defined state transitions, but the workflow model allows for deviations in terms of externally triggered state changes. Thus, the OLCs keep maintainable as they do not have to contain all deviations possible at runtime and business users are enabled to influence the actual process execution by manipulating the processed business objects and

their states. Furthermore, unforeseen events in the business environment can be reflected by state changes and have direct impact on the process execution. The workflow model can be derived in an automatic transformation of an (composite) OLC.

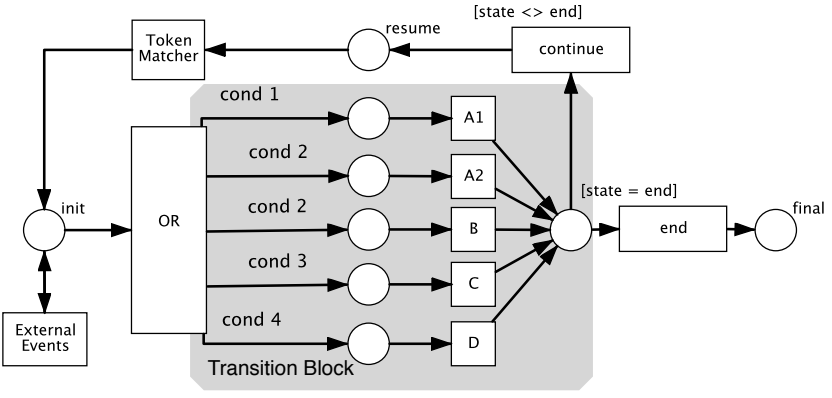


Fig. 2. Overview of the workflow model

We define the workflow model using the Coloured Petri nets formalism as it is illustrated in Fig. 2. The *Transition Block* contains all implementations of the activities performed on an object, i.e., transitions of the OLC. These transitions can be executed completely independent from each other. Tokens running through the net carry a variable that represents a current substate of the process, e.g. a process starts with one token in the place *init* carrying the initial state. According to the substate of a token, the transition *OR* enables transitions in the *Transition Block*. Each outgoing arc of *OR* has an expression, that specifies the substates where the respective transition becomes enabled. If multiple arc expressions are true, all respective transitions are enabled and executed concurrently. Basically, substates correspond to places in the OLC. A place in an OLC is either marked with zero or one token. Thus, a token in the workflow model carrying a value x indicates that in the current marking of the OLC there would be a token in place x. A major intention of the workflow model is to separate control-flow restrictions from the actual execution of transitions. Therefore, we have to mind concurrency and exclusive decisions in the transformation from an OLC to a workflow model.

Concurrency. The concurrent execution of transitions in the workflow model is realized by enabling these transitions at the same substate carried by a token. The synchronization of tokens is also realized independent from the actual execution of transitions in order to keep the model flexible. Synchronization is realized by the *Token Matcher*. It is a subnet, that is entered by all tokens carrying a substate that requires synchronization. Tokens are matched pairwise.

Synchronization of three or more tokens is realized in multiple steps. To be able to keep track of such complex synchronizations, additional substates for tokens are introduced during the transformation.

Exclusive decisions. Exclusive decisions need special handling as the structure of the workflow does not allow for such decisions. Transitions, that are enabled at same substate are executed in parallel. Therefore, exclusive decisions become explicit transitions during the transformation. They occur in the *Transition Block* and produce a token with a substate that indicates which of the exclusive transitions has to be executed. Accordingly, each of the exclusive transitions has a unique substate as its precondition (i.e. arc expression).

External Events. One important requirement for the execution of design processes is the ability to handle unforeseen events. We propose the strict separation of control-flow restrictions and implemented transitions as the solution. An external event causes an arbitrary change of data of an object involved in the process. To be able to continue the process execution, the object must be in a consistent state after the event happened. Therefore, the new object state must occur in the respective OLC and can be translated into valid substates for tokens. Consequently, an external event can be reflected by continuing process execution with a set of tokens carrying the substates that represent the new object state. Note that process continuation after an external event is not equal to a complete rollback. Likely, such an event changes certain properties of an object, but it will not have impact on all data that was created during former process execution. Some transitions might be skipped when continuing process execution, as the data they would produce is already present and still valid. Additionally, an external event during the execution of two concurrent branches might trigger redoing of an activity preceding the parallel split. In this case, either the split and all succeeding activities are performed again or the execution of one branch continues and activities of the other one are executed again, because only data relevant for this branch has changed. To ensure the correct number of tokens in the net at any time, the latter option is traced back to the first one by skipping all activities that have been executed before. We propose that such skipping or redoing of transitions is under the control of domain experts, as an execution engine is in general not able to decide whether existing values are still valid.

4 Relations between Models

The business environment of product design processes is constantly changing. Hence, the development of sufficient support with information systems is an iterative process where changes are very likely. Changes induce the possibility of inconsistencies. Therefore, we propose to relate similar parts among OLCs and and between OLCs and HLMs and present a notion for consistency verification based on these relations. Besides, we investigate OLC composition in this section, which is necessary to express dependencies between different objects.

4.1 Inheritance of Object Life-Cycles

Products are often related to others, they belong to product categories. Members of the same category differ from each other in certain specializations. A methodology for product design processes must reflect such specialization, to express that similar goals are achieved in similar manner. In order to allow for effective handling, specialization dependencies should be reflected when modeling OLCs. Thus, the life-cycle may be specified for a category of products first, while it is later refined for specific product subtypes. We address this demand by introducing a novel notion of inheritance for OLCs. Fig. 3 depicts a parent and a child model, which we will use for illustration purposes. We assume a refinement relation between transitions as indicated by dotted lines, i.e. the transition set $\{A1', A2'\}$ in the child model refines the transition set $\{A\}$ in the parent model. A pair of related transition sets is called correspondence.

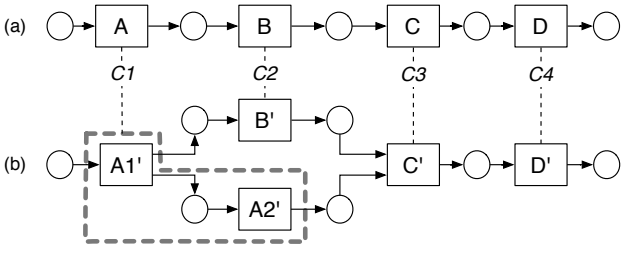


Fig. 3. Specialization of an Object Life-Cycle

Evidently, we do not require such refinements of OLCs to be *hierarchical*. This is motivated by the observation that most sequential orderings of transitions are due to some data dependencies. That is, the preceding transition will provide some data that the succeeding transition requests. During specializing of OLCs, the single transitions might be refined into several transitions representing smaller units of functionality. Now, it is very likely that in the process of refining transitions one will discover potential for optimization in terms of parallelization as the data dependencies hold solely between some of the refined transitions. Fig. 3 illustrates such a scenario. The transition B' depends on the data provided by $A1'$ but is independent from $A2'$. Therefore, it is not necessary to wait until $A2'$ is finished to start B'.

Against this background, existing techniques for the verification of consistency of an inheritance relation between two behavioral models cannot be applied. Those either require refinements to be hierarchical in terms of single-entry-single-exit subgraphs [18] or require that the observable behavior does not change [19,16]. Consequently, the described scenario would be considered to be inconsistent. In order to cope with scenarios as introduced above, our notion of inheritance allows for sequentialization and parallelization. The core idea of the consistency notion is to check whether each ordering of transitions in the parent model, with respect to their correspondences, is reflected by some transitions in

the child model. That is, if there is a transition t_1 in the parent model belonging to a correspondence C_1 that is always preceding a transition t_2 belonging to a correspondence C_2, there must be a transition $t_3 \in C_1$ that always precedes a transition $t_4 \in C_2$ in the child model.

Our inheritance notion is formally grounded on trace semantics. For the model for OLCs introduced above, a complete trace is a firing sequence of Petri net transitions leading from the initial marking to the final marking. In Fig. 3, for instance, the child model induces the traces $A1', B', A2', C', D'$ and $A1', A2', B', C', D'$. To decide consistency regarding to an inheritance relation between two models, consistency between all pairs of correspondences is checked. Consistency of a correspondence pair is decided by comparing all traces of both models. As a correspondence is possibly complex, and therefore, transition sets may have different cardinalities, we partition traces into sub-traces before comparing them.

A partitioning induced by a pair of correspondences is based on a classification of transitions as either being *interleaving*, being part of one of the transition sets without being interleaving, or being not part of any transition set. A transition belonging to either of the transition sets is interleaving if there is another transition belonging to the other transition set and there is at least one reachable marking where these two transitions are enabled concurrently. For our aforementioned example, transitions B' and $A2'$ in the lower model are interleaving transitions, when comparing the correspondences C_1 and C_2.

Fig. 4 illustrates the partitioning of traces for all pairs of correspondences defined in Fig. 3. A sub-trace is labeled like a correspondence, if it contains non-interleaving transitions belonging to the transition set associated to the respective correspondence. Interleaving transitions are labeled by ι and the label of the two respective correspondences. Transitions that are not part of any of the transition sets are neglected. The index t determines the order of the sub-traces. We conclude that both models show equal trace partitionings for the pairs of correspondences (C_1, C_3) and (C_2, C_3). They are sequentially ordered without any interleaving transitions. For the pair (C_1, C_2), however, both models have a different trace partitioning. While the parent model shows a sequential ordering, the child model shows a sub-trace of non-interleaving transitions belonging to C_1 followed by a sub-trace of interleaving transitions belonging to C_1 and C_2.

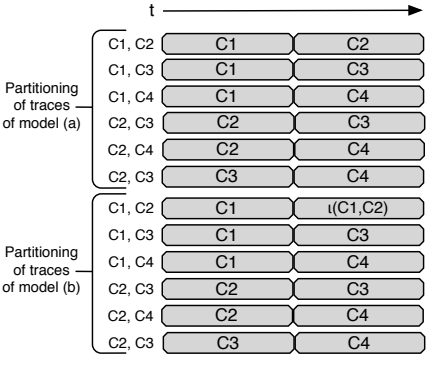


Fig. 4. Trace partitionings of the models in Figure 3

Based on such a trace partitioning we decide whether two correspondences are consistent. Therefore, the partitionings are transformed into a representation

that reflects implied data dependencies only. That is, all interleaving parts are either neglected or replaced by an non-interleaving part belonging to one of the transition sets. An interleaving part is

- hidden if it succeeds a non-interleaving part of one transition set and precedes a non-interleaving part of the other transition set;
- replaced by a non-interleaving part C_2, if it is surrounded by non-interleaving parts of C_1 and vice versa;
- replaced with a non-interleaving part different to the one that precedes the interleaving part, if it is the last part of the trace;
- replaced with a non-interleaving part different to the one it precedes, if it is the first part of the trace.

Traces are considered to be compatible, if they have equal trace partitionings after this transformation. A pair of correspondences is consistent, if for each trace in the parent model, there is a compatible trace in the child model. Two OLCs are in a consistent inheritance relation if all pairs of correspondences are consistent. To ensure, that an object implements the complete behavior of its parent, all transitions of the respective parent model must be part of a correspondence with the child model.

4.2 Composition of Object Life-Cycles

Inheritance of OLCs copes with specialization relations between products. Products may also be related by composition relations. Many products are compositions of several parts with special functionality, which can be developed independently to a certain degree. Such compositions impact also on the respective OLCs and have to be considered when deriving one as a complete system specification for process automation. Furthermore, we propose to specify the behavior of a product in various OLCs that describe different aspects of the products behavior to reduce complexity and to ease maintainability. We distinguish three composition types.

Synchronous. Some activities in a process have impact on multiple objects. For example, the design of intersection points between two independently developed components is one task that affects both components. Execution of such activities triggers state changes in both respective OLCs, and these activities likely have preconditions regarding to all involved objects. Thus, OLCs of these components must be composed not only to derive a complete system specification but also to be able to verify consistent behavior. The composition is realized by identifying and merging transitions of all involved OLCs that represent equal activities.

Asynchronous. In contrast to activities that have impact on multiple objects, other activities require various objects to be in certain states without changing them. That is, certain properties of one component must already be defined because they have impact on the design of other components. In terms of OLCs this means a transition of one model is waiting for the completion of a transition of another model. Consequently, models are composed by connecting these transitions with an additional place.

Alternative Continuation. The OLCs of single objects may become complex, for instance, if objects occur in various contexts where they show different behavior. A common approach to reduce complexity is to define several models that describe different aspects of the behavior of the same object. These models have certain states in common where a context change might happen. Hence, composition is realized by merging all similar places of models describing the same object. Composition by merging places makes sense for models describing the same object only. If one would compose OLCs describing different objects using a place, it would express that two different objects could reach the same state. This is a contradiction to the semantics of a state, as it says that two objects in equal states cannot be distinguished.

Consistent Compositions. Composing OLCs as described before leads to a model that contradicts the definition of an object-life cycle, as it may contain multiple initial and final places. This must be resolved by creating a new initial and a new final place and connecting them with the initial and final places of the OLCs describing the single components. After this transformation, composite OLCs are similar to standalone ones. Thus, the composition must be bounded and safe, as a marking still represents a state and there is no valid interpretation for a place with multiple tokens. Additionally, composite OLCs must always describe the aggregated behavior of all original OLCs and can optionally describe additional behavior in terms of interaction between objects. Consequently, the composite OLC must not contain dead transition, each transition must be part of a trace from the initial to the final marking. Otherwise, the described behavior has changed and the composite OLC is considered to be inconsistent. We have argued before, that syntactically possible deadlocks in OLCs are avoided, because the whole state of an object is considered in exclusive decisions. This argument also holds for OLC compositions. Thus, a consistent composition is relaxed sound.

In sum, OLC composition enables the structured definition of an artifact-centric system specification. A system can easily be extended by adding OLCs describing new aspects and adapted by editing or removing existing OLCs. Consistency verification of both, single models and composite ones, can be applied by checking for relaxed soundness. OLC compositions ease the maintainability of large, constantly changing sets of models and provide methods to derive one system specifications for process execution.

4.3 Relation between High-Level Models and Object Life-Cycles

HLMs and OLCs have very different modeling purposes and, therefore, are different views on the same process. Furthermore, they are at different abstraction levels: while HLMs consist of the most important activities only, object-life cycles give detailed information about the single states and state changes of all involved business objects. Obviously, these models will have many differences, but there are also some similarities, since both views describe behavior of the same process. The similar parts of the models should not contradict in the described behavior. We propose to specify similar parts between HLMs and OLCs

using correspondences. Hence, activities in HLMs correspond to transitions in OLCs. Again, correspondences might be complex, meaning they are defined between sets of activities and sets of transitions. A process likely involves several business objects. That raises the question how to correlate multiple OLCs with one HLM and whether it is allowed that a set of activities in a HLM corresponds to various sets of transitions in different OLCs. This is certainly true, but using the methods described in Section 4.2 these models can be composed to one OLC. Transition sets corresponding to the same activity are indicating synchronization points. Correspondences between the HLM and the composite OLC must be non-overlapping, as the semantics of overlapping correspondences are unclear.

Section 4.1 introduced a consistency notion for OLC inheritance based on complex correspondences. This notion is motivated by data dependencies between activities and can be applied for relating HLMs and OLCs, even though modeling purpose and view on the process are different. In either model type control-flow restrictions for activities are specified. As correspondences identify similar activities, the restrictions defined for them must not contradict. To apply the consistency notion, HLMs have to be translated to Petri net. As previously mentioned, such a mapping for our BPMN subset is given in [14].

5 Implementation

We implemented our approach prototypically and integrated it into the Oryx Project[1]. One part of the project is the Oryx Mashup-Framework. It contains a gadget that allows to define correspondences between models. Based on these correspondences, consistency regarding OLC inheritance can be verified. Fig. 5 depicts a parent (left), a child model (right) and four defined correspondences. The correspondence pair $c3 = (\{C\}, \{C\})$ and $c4 = (\{D\}, \{D\})$ is inconsistent. The parent model implies that C provides some data used in D, but in the child model both transitions are executed concurrently without any dependency. To visualize the inconsistency, all respective transitions have been highlighted. Currently, the implemented algorithm is restricted to sound free-choice Petri nets, i.e. HLMs that can be mapped to sound free-choice Petri nets and sound free-choice OLCs. By this restriction, more efficient calculation of consistency is possible, because of the strong relation between syntax and semantics of sound free-choice Petri nets. Besides the consistency notion, we implemented the transformation from an OLC into a workflow model. To integrate the transformation into the Oryx Editor, a client and a server plugin have been realized. The client plugin offers the functionality to trigger the transformation for the OLC currently opened. The server plugin realizes the transformation and responds with the generated workflow model in a JSON representation. The workflow model can be displayed in Oryx by creating a new Coloured Petri net and importing the JSON. Finally, the model can be exported to CPN Tools[2] for process simulation.

[1] http://www.oryx-project.org
[2] http://cpntools.org/

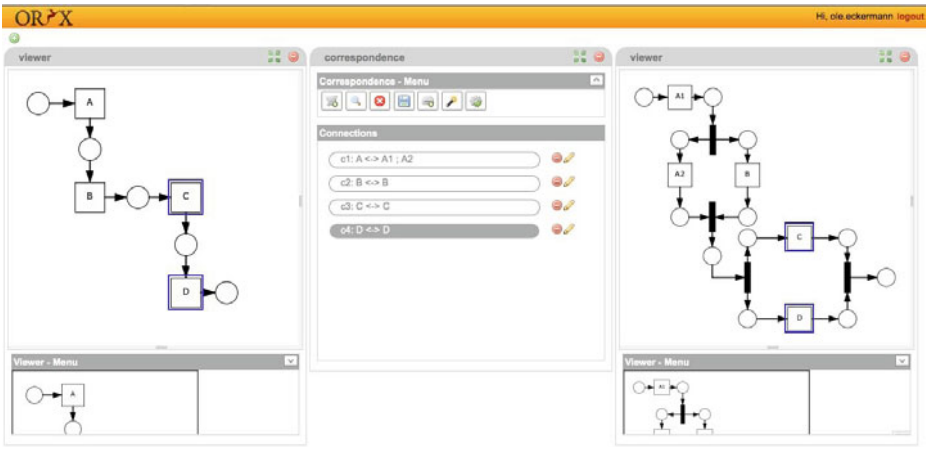


Fig. 5. Transitions regarding to inconsistent correspondences are highlighted

6 Related Work

In recent years, *artifact-centric* design of business processes gained increasing attention, see [10,11]. Following this methodology, the key driver of modeling and execution are business artifacts that are specified by both, an informational model and a life-cycle. Activities operate on artifacts and are responsible for state changes. In contrast to such an integrated view, we propose a methodology containing both process models and OLCs that are interrelated.

Moreover, our approach goes beyond the existing work by defining OLC hierarchies and compositions. To this end, we introduced a notion of inheritance for OLCs that is related to work on behavior inheritance [19,16]. The latter builds on the notion of *branching bisimilarity* and adapts it to the setting of partially corresponding models. For two behavioral models, nodes that are without counterpart are either be *blocked* or *hidden* when assessing behavioral equivalence. The notion of inheritance introduced in this paper is much weaker than the notions of behavior inheritance. We argue that specialization of objects may be non-hierarchical, so that potential sequentialization or parallelization of activities calls for a more relaxed notion of inheritance.

Regarding the composition of OLCs, our notion for synchronous composition is related to work on the composition of UML state machines [20]. Asynchronous composition has been investigated in the work on *proclets* [21]. Finally, the composition of alternative continuations is related to scenario-based modeling using *oclets* [22]. Oclets specify intended behavior, while anti-oclets are used to express forbidden behavior under certain preconditions. A complete process description is derived by composing oclets at runtime. A different approach aiming for more flexibility during process execution is the concept of pockets of flexibility [23]. These approaches mainly concentrate on reducing complexity of modeling flexible (parts of) processes. We enable flexibility by allowing for not-specified execution sequences that are caused by unforeseen external events.

Finally, techniques for adaptive process management are also related to our work. Adaptive process management has been studied in the ADEPT project [7] – a process management system which is able to handle changes during runtime. The creation of consistent process structures by OLC composition is investigated in [24]. They further discuss changes in the process structure in terms of adding/deleting OLCs or dependencies between them. While these approaches aim for easy and consistent adaption during runtime, we rather focus on flexibility. Therefore, we consider them to be complementary.

7 Conclusion

In this paper, we presented an artifact-driven methodology to automate innovative product design processes. We propose two modeling perspectives: high-level models and object life-cycles. Our contribution here is a novel consistency notion that is insensitive to control-flow structures and relies on data dependencies. The notion can be applied to ensure consistency between both perspectives and among object life-cycles with non-hierarchical refinements.

Furthermore, we presented methods for object life-cycle composition to derive a complete system specification. These methods ease adaption, extension and maintainability of complex and constantly changing processes. Finally, we introduced a workflow model for process execution. The novel structure of this model enables rich flexibility by determining control-flow purely state-based. The workflow model is derived in an automatic transformation from an object life-cycle into a Coloured Petri net.

In future research, we aim to extend our approach to instance correlation. In product design it is common, that starting from one proposal a large set of proposals with slightly different values for certain properties is created during the design process. As these proposals do have a lot in common, they are not handled isolated and run as a set through the process. Certain activities might handle them equal to a single instance, i.e. the manipulated data is equal for all instances, others will handle them as a list, for example approval tasks where certain proposals are rejected and others are accepted.

References

1. Dumas, M., van der Aalst, W.M., ter Hofstede, A.H.: Process-aware information systems: bridging people and software through process technology. John Wiley & Sons, Inc., New York (2005)
2. Weske, M.: Business Process Management – Concepts, Languages, Architectures. Springer-Verlag New York, Inc., Secaucus (2007)
3. Leymann, F., Roller, D.: Production Workflow: Concepts and Techniques. Prentice-Hall, Englewood Cliffs (1999)
4. Alves, A., et al.: Web Services Business Process Execution Language Version 2.0. Technical report, OASIS (January 2007)
5. Hollingsworth, D.: The Workflow Reference Model. Technical report, WFMC (January 1995)

6. Lenz, R., Blaser, R., Beyer, M., Heger, O., Biber, C., Bäumlein, M., Schnabel, M.: It support for clinical pathways - lessons learned. In: Volume, M. (ed.) MIE. Studies in Health Technology and Informatics, vol. 124, pp. 645–650. IOS Press, Amsterdam (2006)
7. Dadam, P., Reichert, M.: The ADEPT project: a decade of research and development for robust and flexible process support. Computer Science - R&D 23(2), 81–97 (2009)
8. Reijers, H.A., Rigter, J.H.M., van der Aalst, W.M.P.: The case handling case. Int. J. Cooperative Inf. Syst. 12(3), 365–391 (2003)
9. van der Aalst, W.M., Weske, M., Grünbauer, D.: Case handling: A new paradigm for business process support. Data and Knowledge Engineering 53, 2005 (2005)
10. Bhattacharya, K., Hull, R., Su, J.: A data-centric design methodology for business processes. In: Handbook of Research on Business Process Modeling, ch. 23, pp. 503–531 (2009)
11. 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)
12. Yourdon, E.: Modern structured analysis. Yourdon Press, Upper Saddle River (1989)
13. Luftman, J., Papp, R., Brier, T.: Enablers and inhibitors of business-IT alignment. Communications of the AI 1(3) (1999)
14. Lohmann, N., Verbeek, E., Dijkman, R.M.: Petri net transformations for business processes - a survey. T. Petri Nets and Other Models of Concurrency 2, 46–63 (2009)
15. Preuner, G., Schrefl, M.: Observation consistent integration of views of object life-cycles. In: Embury, S.M., Fiddian, N.J., Gray, W.A., Jones, A.C. (eds.) BNCOD 1998. LNCS, vol. 1405, p. 32. Springer, Heidelberg (1998)
16. Basten, T., van der Aalst, W.M.P.: Inheritance of behavior. Journal of Logic and Algebraic Programming 47(2), 47–145 (2001)
17. Dehnert, J., van der Aalst, W.M.P.: Bridging the gap between business models and workflow specifications. Int. J. Cooperative Inf. Syst. 13(3), 289–332 (2004)
18. Brauer, W., Gold, R., Vogler, W.: A survey of behaviour and equivalence preserving refinements of petri nets. In: Rozenberg, G. (ed.) APN 1990. LNCS, vol. 483, pp. 1–46. Springer, Heidelberg (1991)
19. Schrefl, M., Stumptner, M.: Behavior-consistent specialization of object life cycles. ACM Trans. Softw. Eng. Methodol. 11(1), 92–148 (2002)
20. Ryndina, K., Küster, J.M., Gall, H.C.: Consistency of business process models and object life cycles. In: Auletta, V. (ed.) MoDELS 2006. LNCS, vol. 4364, pp. 80–90. Springer, Heidelberg (2007)
21. van der Aalst, W.M., Barthelmess, P., Ellis, C., Wainer, J.: Workflow modeling using proclets. In: Cooperative Information Systems. In: Scheuermann, P., Etzion, O. (eds.) CoopIS 2000. LNCS, vol. 1901, pp. 198–209. Springer, Heidelberg (2000)
22. Fahland, D.: Oclets – scenario-based modeling with petri nets. In: Franceschinis, G., Wolf, K. (eds.) PETRI NETS 2009. LNCS, vol. 5606, pp. 223–242. Springer, Heidelberg (2009)
23. Sadiq, S.K., Sadiq, W., Orlowska, M.E.: Pockets of flexibility in workflow specification. In: Kunii, H.S., Jajodia, S., Sølvberg, A. (eds.) ER 2001. LNCS, vol. 2224, p. 513. Springer, Heidelberg (2001)
24. 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)

Continuous Planning for Solving Business Process Adaptivity

Andrea Marrella and Massimo Mecella

Dipartimento di Informatica e Sistemistica Antonio Ruberti,
SAPIENZA - Universitá di Roma, Italy
{marrella,mecella}@dis.uniroma1.it

Abstract. Process Management Systems (PMSs, aka Workflow Management Systems – WfMSs) are currently more and more used as a supporting tool to coordinate the enactment of processes. In real world scenarios, the environment may change in unexpected ways so as to prevent a process from being successfully carried out. In order to cope with these anomalous situations, a PMS should automatically adapt the process without completely replacing it. In this paper, we propose a technique, based on continuous planning, to automatically cope with unexpected changes, in order to modify only those parts of the process that need to be changed/adapted and keeping other parts stable. We also provide a running example that shows the practical applicability of the approach.

Keywords: Processes, Adaptivity, Continuous Planning, Process Management Systems.

1 Introduction

Process Management Systems (PMSs, aka Workflow Management Systems) [1] are applied to support and automate process enactment, aiming at increasing the efficiency and effectiveness in its execution. Classical PMSs offer good process support as long as the processes are structured and do not require much flexibility. In the last years, the trade-off between *flexibility* and *support* has become an important issue in workflow technology [2]. If on the one hand there is a desire to control processes and avoid incorrect executions of the processes, on the other hand users want flexible processes that do not constraint them in their action. A recent open research question concerns how to tackle scenarios characterized by being very dynamic and subject to higher frequency of unexpected contingencies than classical scenarios, e.g., a scenario for emergency management. There, a PMS can be used to coordinate the activities of emergency operators within teams. Figure 1 shows a (slightly simplified) example of a possible scenario for the aftermath of an earthquake. The process depicts some actors that assess an area for dangerous partially-collapsed buildings. Meanwhile others are giving first aid to the injured people and filling in a questionnaire. In such a context, the PMS must provide an high degree of both support and flexibility. Hence, if

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 118–132, 2011.

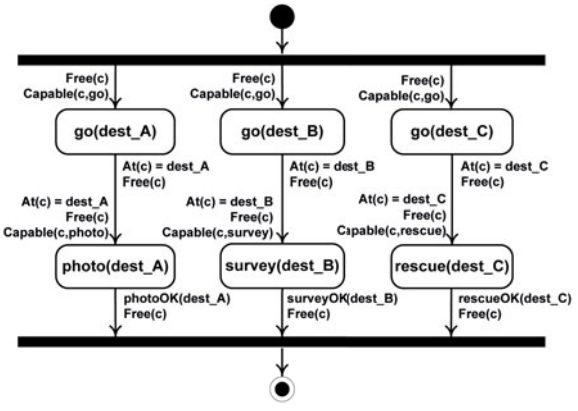


Fig. 1. A possible process to be carried on in disaster management scenarios

on the one hand it should "drive" each actor along the control flow, by guaranteeing that each task in the process is executed in the correct order and with a proper termination, on the other side it should automatically react to exceptions by adapting the process without completely replacing it. This paper proposes a novel approach, in which *continuous planning* [3] is used to improve the degree of *automatic* adaptation in PMSs. The technique, which constitutes an interesting application of non-classical planning, is able to automatically adapt processes without explicitly defining handlers/policies to recover from exogenous events. In order to describe our approach, we make use of a declarative model named SmartPM. Our model allows to define logical constraints and provides a proper execution engine that manages the process routing and decides which tasks are enabled for execution, by taking into account the control flow and the value of variables. Once a task is ready for being assigned, the engine is also in charge of assigning it to a proper service (which may be a human actor, a robot, a software application, etc.). In contrast with other approaches that use planning to handle adaptivity [4,5], our technique provides two interesting features in recovering failures : *(i)* it modifies only those parts of the process that need to be adapted by keeping other parts stable; *(ii)* it is a non-blocking technique; it does not stop directly any task in the main process during the computation of the recovery process. The rest of the paper is organized as follows. Section 2 covers the state of the art in adaptivity in PMSs and relevant results in planning. Sections 3 and 4 introduce the basic preliminary concepts, whereas Sections 5 and 6 illustrate both the general framework and the technique for automatically adapting processes. Finally, Section 7 concludes the paper.

2 Related Works

The approach proposed in this paper tackles the problem of automatic adaptivity in PMSs by using techniques devised from the planning community. Adaptivity

Table 1. Features provided by the leading PMSs to manage adaptation

Product	Manual	Pre-planned	Unplanned
YAWL [6]		✓	
DECLARE [2]		✓	
OPERA [7]	✓	✓	
ADEPT2 [8]	✓		
ADOME [9]	✓		
AgentWork [10]	✓		
ProCycle [11]	✓		
WASA [12]	✓		
SmartPM + Continuous Planning			✓

in PMSs concerns the capability to face *exceptional changes*, which are characterized by events, foreseeable or unforeseeable, during the process instance executions which may require instances to be adapted in order to be carried out. There are two ways to handling exceptional events: *manual* (once events are detected, a responsible person, expert on the process domain, modifies manually the affected instances) or *automatic* (when exceptional events are sensed, PMS is able to change accordingly the schema of affected instances in a way they can still be completed). In the range of automatic adaptation, we can distinguish between two further categories: *pre-planned* adaptation (i.e., for each kind of failure that is envisioned to occur, a specific contingency process is defined a priori) and *unplanned* adaptation. In the latter case, process schemas are defined as if failures could never occur; there is a monitor which is continuously looking for the occurrence of failures. When some of them occur, the process is automatically adapted to mitigate the effects. The difference with the pre-planned adaptation consists in that there exist no pre-planned policies, but the policy is built on the fly for the specific occurrence. Over the years, a multitude of adaptive PMSs (either commercial or research proposals/prototypes) have been developed. Table 1 compares the degree of adaptability to exceptional changes that is currently provided by the leading PMSs (either commercial or research proposals/prototypes). Among them, interesting approaches are ProCycle [11] and ADEPT2 [8]. The first uses a case-based reasoning approach to support adaptation of workflow specifications to changing circumstances. Case-based reasoning (CBR) is the way of solving new problems based on the solutions of similar past problems: users are supported to adapt processes by taking into account how previously similar events have been managed. However, adaptation remains manual, since users need to decide how to manage the events though they are provided with suggestions. ADEPT2 features a check of "semantic" correctness to evaluate whether events can prevent processes from completing successfully. But the semantic correctness relies on some semantic constraints that are defined manually by designers at design-time and are not inferred, e.g., over pre- and post-conditions of tasks. Pre-planned approaches to exceptional changes (a.k.a. exceptions) are often based on the specification of exception handlers and compensation flows [7], with the challenge that in many cases the compensation

cannot be performed by simply undoing actions and doing them again. Our approach is complementary with regard to this literature, and leverages on it for dealing with exceptional changes that can be pre-planned. The novelty is that we propose, in addition to incorporating the previous techniques in a PMS, also to consider automatic adaptation to unplanned exceptions.

2.1 Planning Algorithms

Planning systems are problem-solving algorithms that operate on explicit representations of states and actions. The standard representation language of classical planners is known as the Planning Domain Definition Language [13] (PDDL); it allows to formulate a problem through the description of the initial state of the world, the description of the desired goal and a set of possible actions. An *action definition* defines the condition under which an action can be executed, called *pre-conditions* and its effects on the state of the world, called *effects*. The set of all action definitions represents the *domain* of the planning problem. A planner that works on such inputs generates a sequence of actions (the *plan*) that leads from the initial state to a state fulfilling the goal. The code in Figure 2b depicts the PDDL representation of the task $go(i)$ (it is the first task defined in each branch of the process in Figure 1) that instructs a service c to move towards the destination denoted by i. *Continuous Planning* [3] refers to the process of planning in a world under continual change, where the planning problem is often a matter of adapting to the world when new information is sensed. A continuous planner is designed to persist indefinitely in the environment. Rather than thinking of the planner and execution monitor as separate processes, one of which passes its results to the other, we can think of them as a single process. In order to validate our approach, we make use of a continuous planner working on top of UCPOP planner [14].

3 Preliminaries

In this paper, we use the situation calculus (SitCalc) to formalize adaptation in PMSs. The SitCalc is a second-order logic formalism designed for representing and reasoning about dynamic domains [15]. In the SitCalc, a dynamic world is modeled as progressing through a series of *situations* as a result of various *actions* being performed. A situation represents a history of actions occurred so far. The constant S_0 denotes the initial situation, and a special binary function symbol $do(a, s)$ denotes the next situation resulting from the performance of action a in situation s. Conditions whose truth value may change are modeled by means of *fluents*. Technically, these are predicates taking a situation term as their last argument. Fluents may be thought of as "properties" of the world whose values may vary across situations. Changes in fluents (resulting from executing actions) are specified through *successor state axioms*. In particular for each fluent F we have a successor state axioms as follows:

$$F(\overrightarrow{x}, do(a, s)) \Leftrightarrow \Gamma_F(\overrightarrow{x}, a, s) \tag{1}$$

Table 2. IndiGolog constructs

Construct	Meaning
a	A primitive action.
$\sigma?$	Wait while the σ condition is false.
$(\delta_1; \delta_2)$	Sequence of two sub-programs δ_1 and δ_2.
$proc\ P(\overrightarrow{v})\ \delta$	Invocation of a IndiGolog procedure δ passing a vector $\overrightarrow{v}$ of parameters.
$(\delta_1 \vert \delta_2)$	Non-deterministic choice among (sub-)program δ_1 and δ_2.
$if\ \sigma\ then\ \delta_1\ else\ \delta_2$	Conditional statement: if σ holds, δ_1 is executed; otherwise δ_2.
$while\ \sigma\ do\ \delta$	Iterative invocation of δ.
$(\delta_1 \parallel \delta_2)$	Concurrent execution.
δ^*	Non-deterministic iteration of program execution.
$\pi a.\delta$	Non-deterministic choice of argument a followed by the execution of δ.
$\langle \sigma \rightarrow \delta \rangle$	δ is repeatedly executed until σ becomes false, releasing control to anyone else able to execute.
$send(\Upsilon, \overrightarrow{v})$	a vector $\overrightarrow{v}$ of parameters is passed to an external program Υ.
$receive(\Upsilon, \overrightarrow{v})$	a vector $\overrightarrow{v}$ of parameters is received by an external program Υ.

where $\Gamma_F(\overrightarrow{x}, a, s)$ is a formula with free variables fully capturing the truth-value of fluent F on objects $\overrightarrow{x}$ when action a is performed in situation s. Besides successor state axioms, SitCalc is characterized by *action precondition axioms*, which specify whether a certain action is executable in a situation. Action precondition axioms have the form:

$$Poss(a, s) \Leftrightarrow \Pi_a(s) \tag{2}$$

where the formula $\Pi_a(s)$ defines the conditions under which the action a may be performed in the situation s. In order to control the execution of actions we make use of high level programs, expressed in Golog-like programming languages. In particular we focus on IndiGolog [16], a programming language for autonomous agents that sense their environment and act as they operate. The programmer provides a high-level nondeterministic program involving domain-specific actions and tests to perform the tasks. The IndiGolog interpreter reasons about the preconditions and effects of the actions in the program to find a legal terminating execution. To support this, the programmer provides a SitCalc *theory*, that is a declarative specification of the domain (i.e., primitive actions, preconditions and effects, what is known about the initial state) in the situation calculus. IndiGolog is equipped with standard imperative constructs (e.g., sequence, conditional, iteration, etc.) as well as procedures and primitives for expressing various types of concurrency and prioritized interrupts. The Table 2 summarizes the constructs of IndiGolog used in this work. Basically, these constructs allow to define every well-structured process as defined in [17]. Let's focus on the interrupt construct:

$$\langle\ \sigma \rightarrow \delta\ \rangle \stackrel{\text{def}}{=} \textbf{while}\ Interrupts\_running\ \textbf{do}$$
$$\textbf{if}\ \sigma\ \textbf{then}\ \delta\ \textbf{else}\ false\ \textbf{endIf}$$
$$\textbf{endWhile}$$

To see how this works, first assume that the special fluent $Interrupts\_running$ is identically true. When an interrupt $\langle \sigma \rightarrow \delta \rangle$ gets control from higher priority processes, suspending any lower priority processes that may have been advancing, it repeatedly executes δ until σ becomes false. Once the interrupt body

δ completes its execution, the suspended lower priority processes may resume. The control release also occurs if σ cannot progress (e.g., since no action meets its precondition). The interrupt construct allows IndiGolog to provide a formal notion of *interleaved planning, sensing, and action*. Roughly speaking, an *online execution* of a program finds a next possible action, executes it in the real world, obtains sensing information afterward, and repeats the cycle until the program is finished. The fact that actions are quickly executed without much deliberation and sensing information is gathered after each step makes the approach realistic for dynamic and changing environments. Finally, IndiGolog provides flexible mechanisms for interfacing with other programming languages such as `Java` or `C`, and for socket communication. For our convenience, we have defined here two more abstract constructs to send/receive parameters as well as values with external programs, defined out of the range of the IndiGolog process. For more details about the communication between IndiGolog and external programs, we refer the reader to [16].

4 Process Formalization in Situation Calculus

When using IndiGolog for process management, we take tasks to be predefined sequences of actions and processes to be IndiGolog programs. The monitoring of the process execution is in charge of SmartPM, our declarative PMS deployed by using the IndiGolog interpreter. It drives the task assignment to services involved in the process execution and repairs the process if it is invalidated. To denote the various objects of interest, we make use of the following domain-independent predicates (that is, non-fluent rigid predicates):

- $Service(c)$: c is a service;
- $Task(t)$: t is a task;
- $Capability(b)$: b is a capability;
- $Provides(c, b)$: service c provides the capability b;
- $Requires(t, b)$: task t requires the capability b.

To refer to the ability of a service c to perform a certain task t, we introduce the following abbreviation:

$$Capable(c, t) \stackrel{\text{def}}{=} \forall b.Requires(t, b) \Rightarrow Provides(c, b). \tag{3}$$

That is, service c can carry out a certain task t iff c provides all capabilities required by the task t. The life-cycle of a task involves the execution of four basic actions:

- $assign(c, t, i, p)$: a task t with input i is assigned to a service c. p denotes the *expected output* that t is supposed to return if its execution is successful;
- $start(c, t)$: service c is notified to start task t;
- $ackCompl(c, t)$: service c acknowledges of the completion of task t;
- $release(c, t, i, p, q)$: service c releases task t, executed with input i and expected output p, and returns an output q.

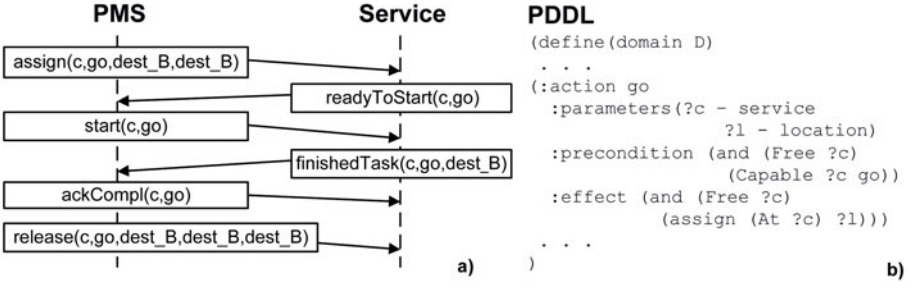


Fig. 2. a) The life cycle and b) the PDDL representation of the task *go*

The terms i, p and q denote arbitrary sets of input/output, which depend on the specific task. The actions performed by the process need to be "complemented" by other actions executed by the services themselves. The following are used to inform the PMS engine about how tasks execution is progressing:

- *readyToStart*(c, t): service c declares to be ready to start performing task t;
- *finishedTask*(c, t, q): service c declares to have completed the execution of task t with output q.

Figure 2a depicts the protocol for a successful execution of a generic task $go(i)$, that instructs a service c to move towards the destination denoted by $i = dest_B$. Note that we suppose to work with domains in which services, tasks, input and output parameters are finite. The formalization of processes in IndiGolog requires two distinct sets of fluents. A first set includes those "engine fluents" that the PMS uses to manage the task life-cycle of processes (for the sake of brevity, here we omit their specification). The second set concerns such fluents used to denote the data needed by process instances; their definitions depends strictly on the specific process domain of interest. These "data fluents" can be used to constrain the task assignment, to record the outcome of a task and as guards into the expressions at decision points (e.g., for cycles, conditional statements). So, if X is a process variable meant to capture the outcome of a (specific) task T, then a SitCalc theory shall include a *data fluent* X_φ[1] with the following successor state axiom:

$$\begin{aligned} &X_\varphi(i, do(a, s)) = q \equiv \\ &\quad \big(\exists c, p.\ a = release(c, T, i, p, q)\big) \vee \\ &\quad \big(X_\varphi(i, s) = q\ \wedge \neg\exists c, p, q'.\ a = release(c, T, i, p, q') \wedge (q' \neq q)\big). \end{aligned} \tag{4}$$

The value of X_φ is changed to value q when one of the corresponding tasks finishes with output q. The formalization allows also to define tasks whose associated data fluents can be customized in according to the process needs. For example, one can require some fluents defined for each service c in the formalization. This is the case of the task $go(i)$, just introduced above. The fluent At_φ

[1] Sometimes we use *arguments-suppressed* formulas, i.e., formulas with all arguments suppressed (e.g. X_φ denotes the arguments-suppressed expression for $X_\varphi(i, s)$).

is meant to capture the new position q of the service c in the situation s after the execution of $go(i)$.

$$\begin{aligned} &At_{\varphi}(c, do(a, s)) = q \equiv \\ &\quad \big(\exists i, p.\ a = release(c, go, i, p, q)\big) \vee \\ &\quad \big(At_{\varphi}(c, s) = q\ \wedge \neg\exists i, p, q'.\ a = release(c, go, i, p, q') \wedge (q' \neq q)\big). \end{aligned} \tag{5}$$

As far as it concerns the task assignment, it is driven by the special fluent $Free(c, s)$, which states if a service c is available in situation s for task assignment:

$$\begin{aligned} Poss(assign(c, t, i, p), s) \Leftrightarrow Capable(c, t)\ \wedge \\ Free(c, s) \wedge (X_{\varphi,1} \wedge ... \wedge X_{\varphi,m}). \end{aligned} \tag{6}$$

A task t can be assigned to a service c iff c is *Free* and is *Capable* to execute t. Moreover, values of data fluents $X_{\varphi,j}$ (where j ranges over {1..m}) possibly included in the axiom should be evaluated.

Example 1. Consider the process instance depicted in Figure 1. In order to represent pre- and post-conditions of each task defined in the control-flow, the following data fluents are needed : *(i)* $At_{\varphi}(c, s)$ stores the location in which the service c is located in situation s; *(ii)* $SurveyOK_{\varphi}(d, s)$ is true in situation s if a survey concerning injured people at destination d has been successfully filled and forwarded to the headquarter; *(iii)* $PhotoOK_{\varphi}(d, s)$ is true if the pictures taken in location d are judged as having a good quality; *(iv)* $RescueOK_{\varphi}(d, s)$ is true if injured people in area d have been supported through a medical assistance. ■

5 General Framework

Before a process starts its execution, the PMS takes the initial context from the real environment and builds the knowledge base corresponding to the initial situation S_0. In S_0 every service is assumed as free. As far as data fluents, their initial value should be defined manually at design-time, since they depend on the specific domain. The PMS also builds an IndiGolog program δ_0 corresponding to the process to be carried on. For simplicity, we consider for the discussion only *well-formed processes* as defined in [17]. Process adaptivity can be seen as the ability of PMS to reduce the gap from the *expected reality* – the (idealized) model of reality that is used by the PMS to deliberate – and the *physical reality*, the real world with the actual values of conditions and outcomes. Roughly speaking, the PMS should be able to find a recovery process δ_h that repairs δ_0 and remove the gap between the two kinds of reality. A first approach in this direction was developed in [18]. That approach synthesizes a linear process δ_h (i.e., a process constituted only by a sequence of actions) inserted at a given point of the original process – exactly the point in which the deviation is identified. In more details, let's assume that the current process is $\delta_0 = (\delta_1; \delta_2)$ in which δ_1 is the part of the process already executed and δ_2 is the part of the process which remain to be executed when a deviation is identified. Then the technique devised in [18] synthesizes a linear process δ_h that deals with the

deviation; the adapted process is $\delta'_0 = (\delta_1; \delta_h; \delta_2)$. However, whenever a process needs to be adapted, every running task is interrupted, since the "repair" sequence of actions $\delta_h = [a_1, \ldots, a_n]$ is placed before them. Thus, all the branches can only resume execution after the repair sequence has been executed. A slight improvement to the last approach was devised in [19], where the technique is refined by "assuming" concurrent branches as independent (i.e., neither working on the same variables nor affecting some conditions). If independent, it allows to automatically synthesize a linear recovery process such that it affects only the branch interested in the deviation. Hence, if the current process is $\delta_0 = (\delta_1 || \delta_2)$ and the branch δ_2 breaks, the approach proposed in [19] synthesizes a recovery process δ_h such that the adapted process is $\delta'_0 = (\delta_1 || (\delta_h; \delta_2))$. Note that also this last approach needs to block the execution of the main process δ_0 until the building of the recovery process δ_h is completed.

The technique proposed in this paper tries to overcome the above limitations by introducing a *non-blocking* repairing technique. The idea is to build the recovery procedure δ_h *in parallel* with the execution of the main process δ_0, avoiding to stop directly any task in the process. Once ready, δ_h will be inserted as a new branch of δ_0 and will be executed in concurrency with every other task. Let's now detail how the proposed technique works. We start by formalizing the concepts of *physical reality* and *expected reality*.

Definition 1. *A physical reality $\Phi(s)$ is the set of all data fluents $X_{\varphi,j}$ (where j ranges over $\{1..m\}$) defined in the SitCalc theory. Hence, $\Phi(s) = \bigcup_{j=1..m}\{X_{\varphi,j}\}$.*

The physical reality $\Phi(s)$ captures the values assumed by each *data fluent* in the situation s. Such values reflect what is happening in the real environment whilst the process is under execution. However, the PMS must guarantee that each task in the process is executed correctly, i.e., with an output that satisfies the process specification. For this purpose, the concept of *expected reality* $\Psi(s)$ is needed. For each fluent X_φ, we introduce a new *expected fluent* X_ψ that is meant to record the "expected" value of X after the execution of a task T. The successor state axiom for this new fluent is straightforward:

$$\begin{aligned} &X_\psi(i, do(a, s)) = p \equiv \\ &\quad \big(\exists c, q.\ a = release(c, T, i, p, q)\big) \vee \\ &\quad \big(X_\psi(i, s) = p\ \wedge \neg\exists c, p', q.\ a = release(c, T, i, p', q) \wedge (p' \neq p)\big). \end{aligned} \tag{7}$$

It states that in the expected reality a task is *always executed correctly* and forces the value of X_ψ to the value of the expected output p.

Definition 2. *An expected reality $\Psi(s)$ is the set of all expected fluents $X_{\psi,j}$ (where j ranges over $\{1..m\}$) defined in the SitCalc theory. Hence, $\Psi(s) = \bigcup_{j=1..m}\{X_{\psi,j}\}$.*

A recovery procedure is needed if the two realities are different from each other, i.e., some tasks in the process failed their execution by returning an output q whose value is different from the expected output p. Since the PMS has to guarantee that each task is executed correctly, if a discrepancy occurs it derives

a flow of repairing actions that turns the physical reality into the expected reality. Formally, a situation s is known as Relevant - candidate for adaptation - iff :

$$\mathsf{Relevant}(\delta_0, s) \equiv \neg\mathsf{SameState}(\Phi(s), \Psi(s)) \tag{8}$$

Predicate $\mathsf{SameState}(\Phi(s), \Psi(s))$ holds iff the states[2] denoted by $\Phi(s)$ and $\Psi(s)$ are the same. Each task defined in δ_0 affects (or is affected by) only a finite number of fluents. This means that each task is interested only in that fragment of reality it contributes to modify.

Definition 3. *A task T affects a data/expected fluent X iff $\exists c, i, p, q, a$ s.t. $a = release(c, T, i, p, q)$ and $X(i, do(a, s)) \Leftrightarrow \Gamma_X(i, a, s)$ (cf. equation 1). We denote it with $T \triangleright X$.*

Definition 4. *A task T is affected by a data/expected fluent X iff $\exists c, i, p, a$ s.t. $a = assign(c, T, i, p)$ and $X \in \Pi_a(s)$ (cf. equation 2). We denote it with $T \triangleleft X$.*

The two latter definitions allow to state a new further definition of $\Phi(s)$ and $\Psi(s)$, whose range can be limited to a specific task T.

Definition 5. *Given a specific task T, a T-limited physical reality $\Phi|_T(s)$ is the set of those data fluents $X_{\varphi,j}$ (where j ranges over $\{1..m\}$) such that $T \triangleright X_{\varphi,j}$ or $T \triangleleft X_{\varphi,j}$. We denote these fluents as $X_{\varphi|_T}$. Hence, $\Phi|_T(s) = \bigcup_{j=1..m}\{X_{\varphi,j|T}\}$ and $\Phi|_T(s) \subseteq \Phi(s)$.*

Definition 6. *Given a specific task T, a T-limited expected reality $\Psi|_T(s)$ is the set of those expected fluents $X_{\psi,j}$ (where j ranges over $\{1..m\}$) such that $T \triangleright X_{\psi,j}$ or $T \triangleleft X_{\psi,j}$. We denote these fluents as $X_{\psi|_T}$. Hence, $\Psi|_T(s) = \bigcup_{j=1..m}\{X_{\psi,j|T}\}$ and $\Psi|_T(s) \subseteq \Psi(s)$.*

From definitions 5 and 6, the following one stems :

Definition 7. *Let $T_1, ..., T_n$ all tasks defined in the SitCalc theory. A physical (expected) reality $\Phi(s)$ $(\Psi(s))$ is the union of all T-limited physical (expected) realities that hold in situation s : $\Phi(s) = \bigcup_{i=1..n} \Phi|_{T_i}(s)$ $(\Psi(s) = \bigcup_{i=1..n} \Psi|_{T_i}(s))$.*

Now, the predicate Relevant can be refined in a way that focuses on a specific task T:

$$\mathsf{Relevant}_T(\delta_0, s) \equiv \neg\mathsf{SameState}(\Phi|_T(s), \Psi|_T(s)) \tag{9}$$

Our framework is able to capture - and to recover from - two different kinds of task failure. An *internal failure* is related to the failure in the execution of a task, i.e., the task does not terminate, or it is completed with an output that differs from the expected one. Example 2 shows such a case. An *external failure* is represented as an exogenous event e, given in input by the external environment, that forces a set of data fluents to assume a value imposed by the

[2] Given a situation s and a set $\overrightarrow{F}$ of fluents, a $state(\overrightarrow{F}(s))$ is the set composed by the values - in s - of each fluent F_j that belongs to $\overrightarrow{F}$. Hence, $state(\overrightarrow{F}(s)) = \bigcup_{j=1..m}\{F_j\}$ s.t. $F_j \in \overrightarrow{F}$.

event itself. Such a new value could differ from the expected one, by generating a discrepancy between the two realities. In order to capture the effects of e, the process designer has to refine the successor state axiom of those fluents whose value can be affected by e. An example of how to catch an exogenous event is shown in the following section.

Example 2. Consider the process instance depicted in Figure 1. Let's suppose that the PMS assigns the task $go(dest_B)$ to the service *srvc*. Assume that *srvc*, instead to reach $dest_B$, ends the execution of the task *go* in $dest_Z$. Then, after the *release* action is executed, the fluent At_φ takes the value $dest_Z$. But this output does not satisfy the expected outcome. The expected output $p = dest_B$ is stored in the fluent At_ψ; it generates a discrepancy between $\Phi|_{go}(s)$ and $\Psi|_{go}(s)$. This means that the $\mathsf{Relevant}_{go}(\delta_0, s)$ holds, and the main process δ_0 needs to be adapted. ∎

6 The Repairing Technique

We now turn our attention to how adaptation is meant to work in our approach. Before starting the execution of the process δ_0, the PMS builds the PDDL representation of each task defined in the SitCalc theory and sends it to an external planner that implements the POP algorithm. In Figure 3, we show how the PMS has been concretely coded by the interpreter of IndiGolog. This framework can be viewed as a dynamic system in which the PMS continually generates new goals in response to its perceptions about physical reality. The main procedure involves three concurrent programs in priority. At a lower priority, the system runs the actual IndiGolog program representing the process to be executed, namely procedure **Process**. This procedure relies, in turn, on procedure **ManageExecution**, which includes task assignment, start signaling, acknowledgment of completion, and final release. The monitor, which runs at higher priority, is in charge of monitoring changes in the environment and adapting accordingly. The first step in procedure **Monitor** checks whether fluent *RealityChanged* holds true, meaning that a service has terminated the execution of a task or an exogenous (unexpected) action has occurred in the system. Basically, the procedure **Monitor** is enabled when the physical or the expected reality (or both) change. If it happens, the monitor calls the procedure **IndiPOP**, whose purpose is to manage the execution of the external planner by updating its initial states and expected goals according with changes in the two realities. **IndiPOP** first builds the two sets *Start* (the initial state) and *Finish* (the goal), by making them equal respectively to $\Phi(s)$ and $\Psi(s)$. As far as concerns the initial state, it will include, for each task t and service c defined in SitCalc theory, the values of $Capable(c, t)$ and of $Free(c, s)$ in addition to the values of data fluents. Then **IndiPOP** catches the partial plan $plan_p$ (that has the form of a set of partial ordering constraints between tasks; it is empty if no failure has happened yet) built till that moment by the external planner and updates it with the new sets Start and Finish. Such updating finds something about $plan_p$ that needs fixing in according with the new realities. Since $plan_p$ has been built working on old values of the two

```
PROC Main()
1  ⟨RealityChanged ∧ ¬Finished → [Monitor()]⟩.
2  ⟨Recovered ∧ ¬Finished → [UpdateProcess()]⟩.
3  ⟨true → [Process(); finish]⟩.

PROC UpdateProcess()
1  receive(Planner, plan_h).
2  Convert(plan_h, δ_h).
3  Update(δ_0, (δ_0||δ_h)).
4  resetReality.
5  resetRecovery.

PROC Monitor()
1  IndiPOP().
2  resetRealityChanged.

PROC IndiPOP()
1  Start = Φ(s) ∪ {⋃_{i=1..k, j=1..n} {Capable(c_i, t_j)}} ∪ {⋃_{i=1..k} {Free(c_i, s)}}.
2  Finish = Ψ(s).
3  receive(Planner, plan_p).
4  send(RemoveConflicts, [plan_p, Start, Finish]).
5  receive(RemoveConflicts, plan_u).
6  send(Planner, [plan_u, Start, Finish]).

PROC ManageExecution(Task, Input, ExpectedOutput)
1  π(Srvc).assign(Srvc, Task, Input, ExpectedOutput).
2  start(Srvc, Task).
3  ackCompl(Srvc, Task).
4  release(Srvc, Task, Input, ExpectedOutput, RealOutput).
```

Fig. 3. A fragment of the core procedures of the IndiGolog PMS

realities, it is possible that some ordering constraints between tasks are not valid anymore. This causes the generation of some conflicts, that need to be deleted by $plan_p$ through the external procedure **RemoveConflicts**. Basically, **IndiPOP** can be seen as a conflict-removal procedure that revises the partial recovery plan to the new realities. At this point, $plan_u$ (that is, $plan_p$ just updated, i.e., without conflicts) is sent back to the external planner together with the sets Start and Finish. The external planner can now restore its planning procedure. Note that if the predicate Relevant(s) holds, meaning that a misalignment between the two realities exists, the PMS tries to continue with its execution. In particular, every T_i whose T-limited expected reality $\Psi|_{T_i}(s)$ is different from the T-limited physical reality $\Phi|_{T_i}(s)$ could not more proceed with its execution. However, every task T_j not affected by the deviation can advance without any obstacle. Once sent the sets of fluents composing the two realities to the external planner, the monitor resets the fluent *RealityChanged* to *false*, and the control passes to the process of interest (i.e., program **Process**), that may again execute/advance. When the external planner finds a recovery plan that can align physical and expected reality, the fluent *Recovered* is switched to *true* and the procedure **UpdateProcess** is enabled. Now, after receiving the recovery process δ_h from the planner, the PMS updates the original process δ_0 to a new process δ_0' that, respect to its predecessor, has a new branch to be executed in parallel; such branch is exactly δ_h. It contains all that tasks able to repair the physical reality from the discrepancies (i.e., to unblock all that tasks stopped in δ_0 because their preconditions did not hold). Note that when δ_h is merged with the original

process δ_0, the two realities are still different from each others. Therefore, the PMS makes them equal by forcing $\Psi(s)$ to the current value of $\Phi(s)$. This because the purpose of δ_h, after that all recovery actions have been executed, is to turn the current $\Phi(s)$ into $\Psi(s')$, where s' is that situation reached after the execution of recovery actions. Let us now formalize the concept of *strongly consistency* for a process δ_0.

Definition 8. *Let δ_0 be a process composed by n tasks $T_1, .., T_n$. δ_0 is **strongly consistent** iff:*

- *Given a specific task T and an input I, $\nexists c, c', p, p', q, q', a, a'$ s.t. $a = release(c, T, I, p, q) \ \wedge a' = release(c', T, I, p', q') \wedge (p \neq p')$.*
- *$\forall j \in 1..m, \nexists (T_i, T_k)_{i \neq k}$ s.t. $(T_i \triangleright X_{\varphi,j} \wedge T_k \triangleright X_{\varphi,j})$.*

Intuitively, a process δ_0 is strongly consistent if a specific task, executed on a given input, cannot return different values for its expected output; moreover, the above condition holds if do not exist two different tasks that affect the same fluent. For strongly consistent processes, we can state the concept of *goal* :

Definition 9. *Given a strongly consistent process δ_0, composed by n tasks $T_1, ..., T_n$, the goal of δ_0 can be defined as the set of all expected fluents $X_{\psi,j}$ that are affected by $T_1, ..., T_n$. Hence, $Goal(\delta_0) = \{X_{\psi,j}$ s.t. $\exists i_{1..n}.(T_i \triangleright X_{\psi,j})\}$.*

After a recovery procedure δ_h, $Goal(\delta_0) \subseteq Goal(\delta_0 || \delta_h)$, since the recovery procedure can introduce new tasks with respect to the original process δ_0. Anyway, the original $Goal(\delta_0)$ is preserved also after the adaptation procedure.

Theorem 1 (Termination). *Let δ_0 be a strongly consistent process composed by a finite number of tasks $T_1, ..., T_n$. If δ_0 does not contain while and iteration constructs (cf. Table 1), and the number of exogenous events is finite, then the core procedure of IndiGolog PMS terminates.*

We want to underline that the termination cannot be guaranteed if δ_0 contains loops or iterations, since potentially the two realities could indefinitely change. The same is true if the number of exogenous events is unbounded.

Example 3. Suppose that the process depicted in Figure 1 starts its execution and reaches a situation s where some tasks have been completed by returning their expected outputs. In particular, suppose that the left branch of the process has been completely executed, by obtaining $PhotoOK_{\varphi}(dest_A, s) = true$ and $PhotoOK_{\psi}(dest_A, s) = true$, whilst the other tasks are still under execution. We have defined an exogenous event $photoLost(d)$ where d is a specific location. Such an exogenous event models the case when some photos, previously taken in d, get lost (e.g., due to the unwilling deletion of some files). Consequently, if the exogenous event $photoLost(dest_A)$ occurs, its effect is to force $PhotoOK_{\varphi}(dest_A, \overline{s})$ in the new situation $\overline{s}$ to be *false*, whilst $PhotoOK_{\psi}(dest_A, \overline{s})$ continues to hold. This means that $\mathsf{Relevant}(\delta_0, \overline{s}) \equiv \neg\mathsf{SameState}(\Phi(\overline{s}), \Psi(\overline{s}))$ and that the PMS should find a recovery program which restores the previous value for the fluent $PhotoOK_{\varphi}$. For this purpose, the PMS invokes the external planner. While the planner starts to build the recovery process, let us see the case in which,

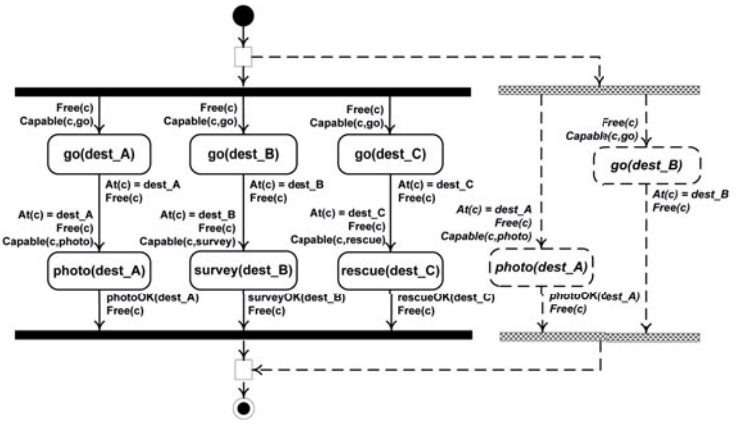


Fig. 4. The process in Figure 1 just fixed with a new repairing branch

in the meanwhile, the task $go(dest_B)$ terminates with a different output by the expected one. In particular, suppose that the service $srvc_2$, that is executing $go(dest_B)$, reaches $dest_Z$ instead of $dest_B$. Hence, we have different values for $At_\varphi(srvc2, \overline{s'})$ and $At_\psi(srvc2, \overline{s'})$. Again, the PMS invokes the external planner by obtaining the partial plan $plan_p$ built till that moment and verifies if it needs to be fixed according with new values of the two realities. If no conflicts are individuated, the PMS sends back $plan_p$ to the planner together with the information about the initial state and the goal, updated to situation $\overline{s'}$. Note that in situation $\overline{s'}$ the task *survey* cannot proceed because one of its preconditions does not hold. When the planner ends its computation, it returns the recovery process δ_h, that can be executed in concurrency with δ_0 (see the right-hand side of Figure 4) by preserving its original goal. ∎

7 Conclusions

In this paper, we advocated the use of a declarative model named SmartPM for automatic process adaptation based on continuous planning. If an unexpected deviation is detected, a recovery process will be built and executed in parallel to the main process. The non-blocking repairing technique enables to reach all goals that would have been reached in the original process. Future works will include an extensive validation of the approach with real collaborative processes and will face the drawbacks provided by the use of continuous planning, such as the risk to introduce data inconsistency when repairing.

References

1. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer, New York (2007)
2. van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative Workflows: Balancing between Flexibility and Support. Computer Science - R&D 23, 99–113 (2009)

3. Brenner, M., Nebel, B.: Continual Planning and Acting in Dynamic Multiagent Environments. Autonomous Agents and Multi-Agent Systems 19, 297–331 (2009)
4. Gajewski, M., Meyer, H., Momotko, M., Schuschel, H., Weske, M.: Dynamic Failure Recovery of Generated Workflows. In: 16th Int. Conf. on Database and Expert Systems Applications, pp. 982–986. IEEE Computer Society Press, Los Alamitos (2005)
5. R-Moreno, M.D., Borrajo, D., Cesta, A., Oddi, A.: Integrating Planning and Scheduling in Workflow Domains. Expert Syst. Appl. 33, 389–406 (2007)
6. Hofstede, A., van der Aalst, W., Adams, M., Russell, N.: Modern Business Process Automation: YAWL and its Support Environment. Springer Publ. Company, Heidelberg (2009)
7. Hagen, C., Alonso, G.: Exception Handling in Workflow Management Systems. IEEE Trans. Soft. Eng. 26(10), 943–958 (2000)
8. Göser, K., Jurisch, M., Acker, H., Kreher, U., Lauer, M., Rinderle, S., Reichert, M., Dadam, P.: Next-generation Process Management with ADEPT2. In: Demonstration Prog. at the Int. Conf. on Business Process Management (2007)
9. Chiu, D., Li, Q., Karlapalem, K.: A Logical Framework for Exception Handling in ADOME Workflow Management System. In: 12th Int. Conf. of Advanced Information Systems Engineering, pp. 110–125 (2000)
10. Müller, R., Greiner, U., Rahm, E.: AGENTWORK: a Workflow System Supporting Rule-based Workflow Adaptation. Data & Knowledge Eng. 51(2), 223–256 (2004)
11. Weber, B., Reichert, M., Rinderle-Ma, S., Wild, W.: Providing Integrated Life Cycle Support in Process-Aware Information Systems. Int. J. of Cooperative Information Systems 18(1), 115–165 (2009)
12. Weske, M.: Formal Foundation and Conceptual Design of Dynamic Adaptations in a Workflow Management System. In: Hawaii Int. Conf. on System Sciences (2001)
13. Mcdermott, D., Ghallab, M., Howe, A., Knoblock, C.A., Ram, A., Veloso, M., Weld, D.S., Wilkins, D.E.: PDDL - The Planning Domain Definition Language. Technical report (1998)
14. Penberthy, S.J., Weld, D.S.: UCPOP: A Sound, Complete, Partial Order Planner for ADL. In: Int. Conf. on the Principles of Knowledge Representation and Reasoning, pp. 103–114 (1992)
15. Reiter, R.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press, Cambridge (2001)
16. De Giacomo, G., Lespérance, Y., Levesque, H.J., Sardina, S.: IndiGolog: A High-Level Programming Language for Embedded Reasoning Agents. In: Multi-Agent Programming, pp. 31–72 (2009)
17. Kiepuszewski, B., ter Hofstede, A.H.M., Bussler, C.: On Structured Workflow Modelling. In: Int. Conf. of Advanced Information Systems Eng., pp. 431–445 (2000)
18. de Leoni, M., Mecella, M., De Giacomo, G.: Highly dynamic adaptation in process management systems through execution monitoring. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 182–197. Springer, Heidelberg (2007)
19. de Leoni, M., De Giacomo, G., Lespèrance, Y., Mecella, M.: On-line Adaptation of Sequential Mobile Processes Running Concurrently. In: ACM Symposium on Applied Computing, pp. 1345–1352 (2009)

Distributed Event-Based Process Execution - Assessing Feasibility and Flexibility

Pieter Hens[1], Monique Snoeck[1], Manu De Backer[1,2,3,4], and Geert Poels[2]

[1] K.U.Leuven, Dept. of Decision Sciences and Information Management, Naamsestraat 69, 3000 Leuven, Belgium
[2] Universiteit Gent, Dept. of Management Information and Operations Management, Tweekerkenstraat 2, 9000 Gent, Belgium
[3] Universiteit Antwerpen, Dept. of Management Information Systems, Prinsstraat 13, 2000 Antwerpen, Belgium
[4] Hogeschool Gent, Dept. of Management and Informatics, Kortrijksesteenweg 14, 9000 Gent, Belgium

Abstract. Processes modeling and execution (with a process engine) are getting more and more incorporated in todays business environments. This movement puts a lot of stress on classical process engines which have to coordinate many process instances simultaneously. Performance degrades quickly as the number of process instances increases, and a single point of failure is introduced by using a central process execution engine. In this paper, we address these challenges by providing a non-intrusive approach to distribute a process flow and have the flow executed by multiple, smaller process engines. We pay special attention to flexibility of the eventual distributed execution, since process change is costly in a distributed environment. We demonstrate the feasibility of our approach by providing an implementation of the transformation and execution architecture, and demonstrate the lower cost of process change that is achieved when using a flexible process runtime architecture.

1 Introduction

Process-aware information systems (PAISs) are becoming more and more integrated in todays business environments [1]. Companies are aware of the running processes in their organization, where they analyze, model and execute these processes. Together with Service Oriented Architectures, these processes can be executed automatically by a process engine. Executing a process logic means coordinating the described work, invoking the correct services, adding tasks to the inbox of task managers, and choosing the correct control flow paths [2].

In classical process execution architectures, one process engine is responsible for the execution of one (designed) process model. However, the higher use and incorporation of processes in todays businesses means that one process engine has to handle a significant amount of process instances simultaneously. This puts a high pressure on the process engine, and performance degrades quickly as the number of process instances increases [3,4]. Alongside *degradation of the*

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 133–147, 2011.

performance, centralized execution also adds a *single point of failure* to the process architecture. Services are distributed and decentralized, but the decision logic and coordination of the workflow is still located at one point. Failure of the coordinator means failure of the entire process, even if the services themselves are still available [4,5].

In this paper we propose a flexible, distributed approach to process execution to overcome the drawbacks of centralized execution. Besides the application services, the process flow (process logic) itself is also distributed in the IT architecture. We look at a distributed approach that has the following features:

- A non-intrusive approach, where the process flow is distributed (split) automatically at deployment time, without interference by the original process modeler.
- The split processes are each executed by a dedicated process engine, which differs both physically and logically from the other split process engines.
- The process runtime architecture is robust and scalable. There is no performance bottleneck or single point of failure.
- Unlike other process distribution approaches, we also focus on a loosely coupled and flexible runtime architecture. The distributed process infrastructure should handle process change [6] (at modeling level, as well as at execution level) with minimal cost.

The paper is structured as follows. We first start with a small example, which is used to explain the concepts throughout the paper. Next, we explain the general idea as well as the advantages of our distributed process execution approach. In section 4, the algorithm to distribute the global process flow is explained, together with a demonstration on how the algorithm can be implemented using BPMN as the process modeling language. Section 5 shows a prototype execution architecture and we continue with an illustration of the possibilities and advantages that can be reaped by using a loosely coupled architecture (Sect. 6). In section 7 we situate our approach in other existing proposals for distributed process execution and end with a conclusion (Sect. 8).

2 Running Example

Figure 1 shows a (BPMN) process model for a pizza delivery company. It involves three parties, a chef who bakes the pizzas and creates, if required, side dishes, a cashier who receives the orders and arranges the payments, and a delivery boy who eventually delivers the pizzas. This simple process incorporates the most frequently used process constructs [7] (simple sequence flows, exclusive and parallel gateways (split and join), lanes and a start and end event), human tasks and a service task. Although the example is kept simple for explanatory purposes, the proposed approach is not limited to these simple examples. Also note that this example omits data-flow considerations. Since this research is focused on the control flow of process execution, we assume data can be transmitted along with the sequence flow [8]. For the organization of data-dependencies in a distributed process architecture we refer to [9].

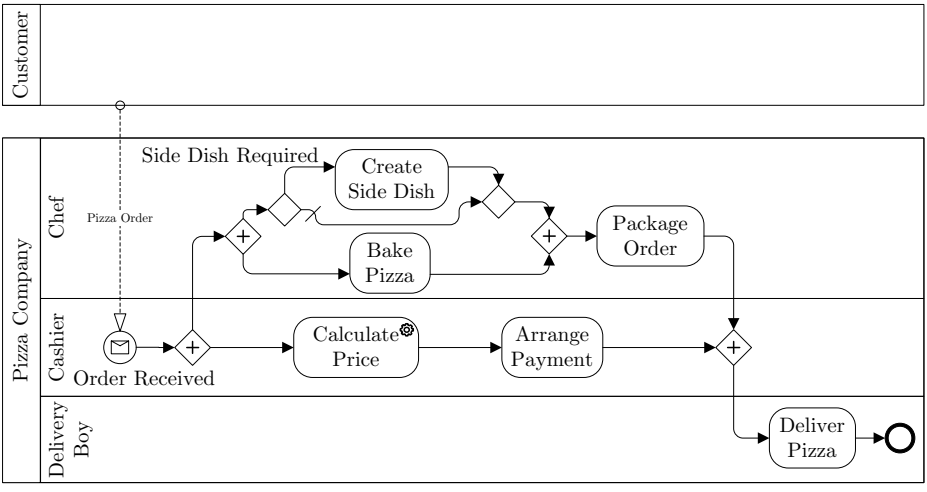


Fig. 1. Pizza Delivery Example

In the pizza company, a process engine is used to execute and control the process flow and a task manager is employed to handle the manual tasks. The process engine adds tasks to the task inbox of the respective performer and the task manager notifies the process engine of any completed tasks. On the other hand, to calculate the price, a (automated) service task is invoked by the process engine, and the service notifies the process engine of its completion (after which the process flow continues).

3 Decentralized Event-Based Orchestration

Figure 2a shows a part of the pizza company's process in the classical centralized approach to process execution. One engine coordinates the process flow (for each process instance) and invokes necessary services distributed in the IT architecture (*PriceCalculator* and *TaskManagers*). As mentioned in the introduction, centralized execution has many drawbacks which include *a single point of failure*, *performance degradation* and *unnecessary network traffic* [4]. To solve these drawbacks, several researchers have proposed solutions to distribute the process logic and use multiple process engines to execute, together, the entire process flow [4,5] (see Fig. 2b). These solutions solve the fundamental problems of central orchestration, but not to a full extent [10]. The distributed (split) execution engines still remain tightly coupled in the process execution architecture. The start of one split engine relies on decisions (invocations) made by others (see the invocation links in Fig. 2b). As explained in [11] this request style of communication creates inflexible IT infrastructures and decreases scalability of the global process architecture.

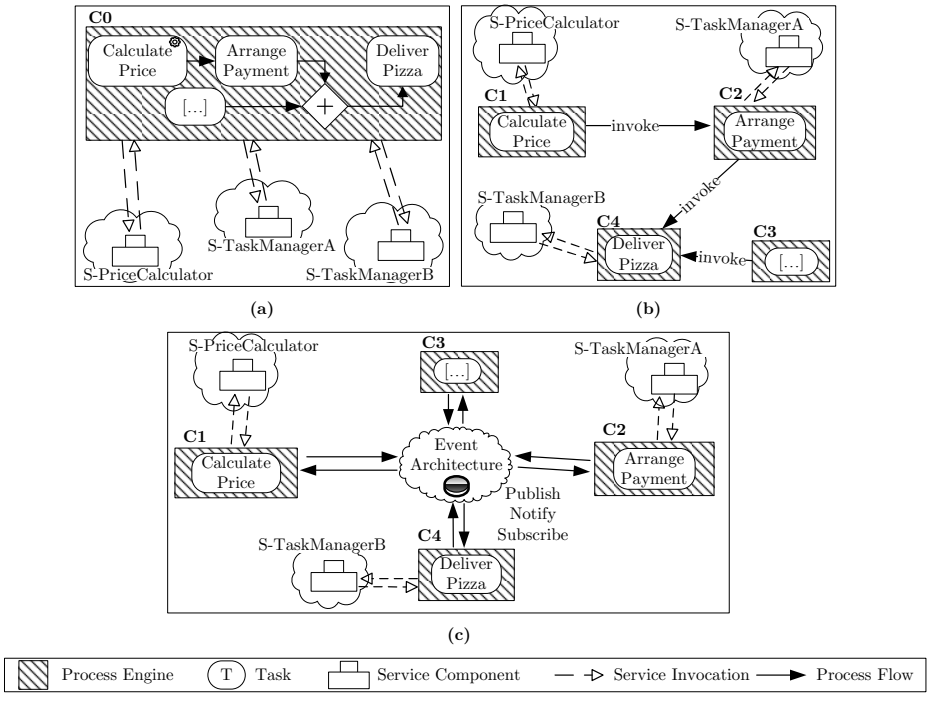


Fig. 2. Centralized, Decentralized and Event-Based Orchestration

To solve the tight coupling, we propose an extension to the distributed process approach, which uses an event-based architecture as the communication paradigm in the distributed process execution [10] (see Fig. 2c). Event based communication is a communication style that uses a publish/subscribe interaction scheme. An event is something that happens, and when an event occurs, a notification of this event occurrence is published in the architecture, where interested parties can receive this event notification. In contrary to a request-based communication style, an event message is non-directed and no expectancies (or SLAs) exist between the sender and *the possible* receiver of an event message [11]. In request-based communication, the responsibility for the next step is located at the caller (the process engine of *CalculatePrice* has the responsibility of invoking the start of *ArrangePayment*), while in event-based communication, this responsibility is located at the callee (the process engine of *ArrangePayment* is itself responsible for starting its execution at the correct time).

Decoupling of interaction partners is the main advantage of using event communication. This decoupling is defined as space decoupling (unawareness of interaction partners), time decoupling (interaction partners don't need to be active at the same time) and synchronization decoupling (asynchronous send and receive) [12]. Together with the switch of responsibility from the caller to the callee, using an event based architecture creates a highly flexible and scalable process execution infrastructure. New pieces of the global process flow, can simply be

added to the process architecture without making any changes to the already running infrastructure (they hold their own starting logic). Note that the supporting entities in an event based architecture (the *cloud* in Fig. 2c) are also distributed and don't add another single point of failure. Many solutions exist that distribute the event based architecture itself [11].

To reap the benefits of event based communication, we need to transform the global process flow to a distributed *event-based* process. The most important part in this transformation is finding the starting rule of each split process. This is described in the following sections.

4 Transformation

To transform a global process into distributed segments, we choose a *task* as the unit of decomposition (the task can also be an embedded subprocess). Each

Definitions
$Process = <T, G, SF>$
with T the set of tasks, G the set of Gateways and SF the set of sequence flows in the process model, with
$SF \subseteq [(T \cup G) \times (T \cup G)]$
$DNFEventRule = \{Conjunction\}$
$Conjunction = <Events, Conditions>$
$Events = \{<id, Signal>\}$
with $id \in \mathbb{N}$ and $Signal$ indicating the occurrence of a happening (e.g. completion of a task)
$Conditions = \{ConditionalExpression\}$
with $ConditionalExpression$ a logical expression
The $\vee$ (OR) operator on two DNFEventRules A and B is defined as follows:
$A \vee B = \{A \cup B\}$
The $\wedge$ (AND) operator on two DNFEventRules A and B is defined as follows:
$A \wedge B = \{X + Y | X \in A, Y \in B\}$
with the + operator on two Conjunctions defined as follows:
$<E_X, C_X> + <E_Y, C_Y> = <E_X \cup E_Y, C_X \cup C_Y>$

split(Process P)
for each Task t $\in$ P **do**
ER = eventRule(t)
create new Process **with**
ER as start rule of the process with
|input places of the process| = |distinct Events $\in$ ER|
t as only task in the process
s = SignalOf(t) as the end/output of the process
$SF = (ER \times t) \cup (t \times s)$
end create
end for each

eventRule (Task t)
$F = \{(x,t) | (x,t) \in SF\}$
$eventRule = \bigvee_{f \in F} eventRule(f)$
eventRule (SF (a,b))
if a = Task **then**
id++
$event = <id, SignalOf(a)>$
$eventRule = \{<\{event\}, \{\}>\}$
else if a = StartOfProcess **then**
$event = <id, a>$
$eventRule = \{<\{event\}, \{\}>\}$
else if a = XOR-Gateway **then**
$F = \{(x,a) | (x,a) \in SF\}$
$eventRule = (\bigvee_{f \in F} eventRule(f)) \wedge \{<\{\}, \{ConditionOn((a,b))\}>\}$
else if a = AND-Gateway **then**
$F = \{(x,a) | (x,a) \in SF\}$
$eventRule = \bigwedge_{f \in F} eventRule(f)$
end if

Fig. 3. Transformation algorithm

task in the global process flow will become a small process itself, with a dedicated process engine. Choosing a task as the unit of decomposition, guarantees a fine grained distribution of the global process flow. The transformation and process executions shown in this paper can easily be extended to allow for other decomposition units, e.g. splitting according to user-defined regions, splitting according to workflow variants [13] or splitting according to the domain a task belongs to [14]. In fact, it suffices to translate a process region to an embedded subprocess to define it as a non-splittable unit of decomposition.

Figure 3 shows the algorithm to split a global process into multiple processes (in $O(n^2)$ time). Each resulting process consists of a starting rule, a task to execute and an (end) event to publish the completion of the task. A starting rule for a split process consists of an event part (the event rule) and a user-defined conditions part (originating from XOR-splits in the global process). Finding the event rule for a split process equals finding, for a specific task, which preceding tasks in the process flow need to be completed before it can start its own execution. The algorithm finds these completion events in a depth-first search in the upward flow in the global process model. The event rule is transcribed as a logical expression in Disjunctive Normal Form, where an element in the expression is a happening in the information system (which we call events). For example, *PackageOrderComplete* `AND` *ArrangePaymentComplete* indicates a rule saying that the split process can start when tasks *PackageOrder* and *ArrangePayment* are completed (notifications are caught indicating the completion of these tasks). In the algorithm in Fig. 3 an event is notated as a tuple $<id, signal>$ with *signal* the event we want to receive (e.g. completion of task *PackageOrder*) and *id* a unique identifier. The id is necessary to make a distinction between two event rules, which have the same logical combination of event types, but have different execution semantics. For example, the distinction between the event rule `((<1,A> AND <1,B>) OR (<1,B> AND <1,C>))` for a split process and the rule `((<1,A> AND <1,B>) OR (<2,B> AND <1,C>))` for another split process is shown in Fig. 4. In the latter event rule (Fig. 4b), *two* input places are enabled by an event B, which enables the possibility of two runs of the task X in the split process. With the first rule (Fig. 4a), only *one* input place is enabled by an event B. The distinction between these two rules is required for the transformation of non-safe process models, where it is possible to have multiple instances of one task, within one process instance.

The second part of a starting rule for a split process consists of user defined conditions originating from XOR-splits. These conditions are in conjunction with the event rule. Only when an event rule evaluates to true AND the respective conditions evaluate to true, then is the task in the split process able to execute. When searching for the completion events for the event rule, any condition encountered on an XOR-gateway is also stored in the starting rule (see Fig. 3).

How a starting rule is transcribed in the resulting split process is dependent on the process language used to describe the split processes. The algorithm in Fig. 3 is kept general, and only gives guidelines on how to build the split processes. Below we give a concrete example of the transformation and the transcription

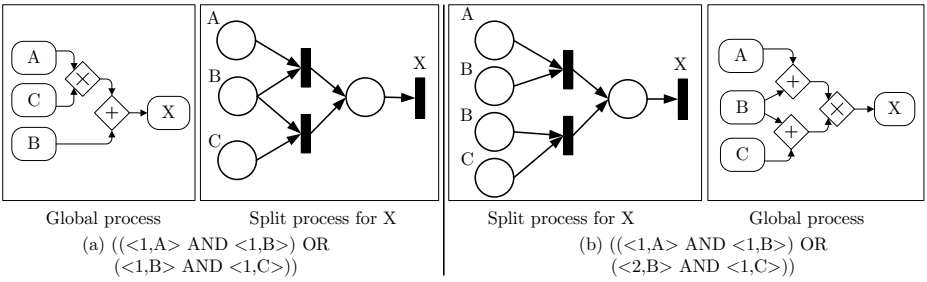


Fig. 4. Use of the identifier element in an event rule

of the starting rule, where we use BPMN2.0 as the language in which the global process as well as the split processes are described.

Algorithmic implementation with BPMN. We implemented the algorithm given in the previous section in the Atlas Transformation Language (ATL) [15]. ATL is a declarative language to describe a transformation of a source model (supported by a meta-model) to a target model. An eclipse plugin is available to create and execute these transformations. We have chosen to make an implementation transforming a BPMN2.0 [8] model to another BPMN2.0 model, where the first model represents the global process and the second the split processes. An advantage of describing the split models in the same language as the global model, is that existing process engines supporting the global model can also execute the distributed process flow, as long as they support communication of events with the publish/subscribe architecture.

As a starting point, we used the *ecore meta-model* of BPMN2.0, available at [16]. Any BPMN model conforming to this meta-model can be used as input for the ATL-transformation (a BPMN Diagram Interchange XML-file [8]). To create the split processes, the ATL-transformation follows the algorithm described in the previous section, where a split process in BPMN is transcribed with the following properties:

- *Signal* events are used as start and end event for the split process. The semantics of a signal event in BPMN conform to the semantics of a notification in an event-based architecture. A throw signal is broadcasted without being directed to one particular process and can hence be caught by any and multiple receiving processes.
- A conjunction in the event rule is represented by multiple *event definitions* within one start event, with the *parallel multiple marker* for that start event set to true (see Fig. 5 for an excerpt of the event BPMN metamodel).
- A disjunction in the event rule is represented by using multiple start events.
- Conditions are placed on the respective sequence flow from the start event to the task.

Figure 6 shows an example of the BPMN representation of the split process for the *Package Order* task. It has the starting rule *((Pizza Baked AND Side*

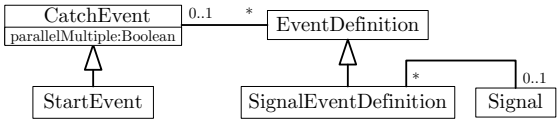


Fig. 5. An exert of the event meta model of BPMN2.0

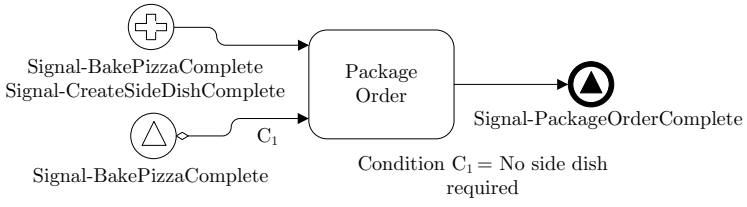


Fig. 6. Example of a resulting split process in BPMN

Dish Created) OR (Pizza Baked, with condition: No side dish required)). Note that the specifications of BPMN2.0 state that a conditional flow (a sequence flow carrying a condition) can't be connected from a start event to a task. We still use this notation as syntactic sugar. To make the model compliant with the BPMN specifications, the model can be easily changed by adding an XOR gateway in between the start event and the task.

The new split processes are stored in an XML-file conforming to the original BPMN2.0 metamodel. Any process engine, or BPMN editor, which supports the BPMN2.0 Interchange format can then be used to open, execute or visualize the resulting file describing the split processes. Figure 7 shows an example of an XML-input file and its resulting transformation, converted with our ATL implementation. In the output file, you'll find for each task a new process description, together with process-wide signal events indicating the completion of each task.

5 Architecture and Process Execution

After the transformation, each split process can be deployed to a dedicated process engine (see Fig. 8). Communication between the process engines happens with a publish/subscribe mechanism. For our prototype execution architecture, we've chosen the Siena wide area event notification service [17]. Siena is a publish/subscribe implementation specifically directed to event subscription and notification in a wide area network, and provides all the necessary routing topology to transmit an event notification from the publisher to the subscriber. By being able to use multiple event services on the wide area network and because of the efficient routing, the single point of failure and performance bottleneck of the central orchestration are solved. Of course, to accomplish the event-based communication, any event architecture can be used, like WS-Notification [18], EVE [19] or the more recently proposed BPEL and WSDL extensions for an

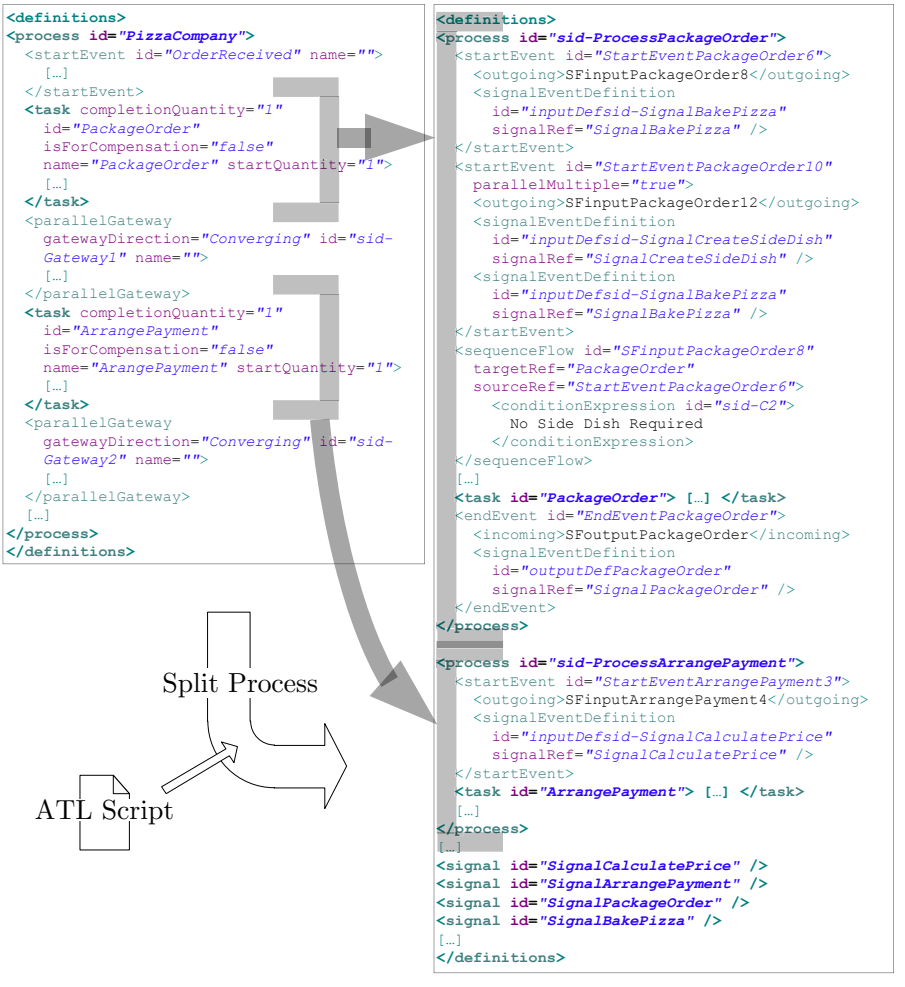


Fig. 7. Transformation of a global BPMN process to an event-based split BPMN process

event driven architecture [20]. A process engine should only be able to communicate with the event dispatchers in the event architecture.

The data payload (content) of an event message in the architecture should minimally consist of two things, one is the indication of the task it represents (e.g. the signal name found in the BPMN file, see Fig. 7), and another is a process instance id, indicating for which (global) process instance an action has been performed. The latter attribute is necessary to not loose the coupling between the process instance and the action performed. The payload of an event message can also be used to distribute any data related to the process execution (e.g. the event indicating the completion of *CalculatePrice* can incorporate the calculated price in the payload of the event message).

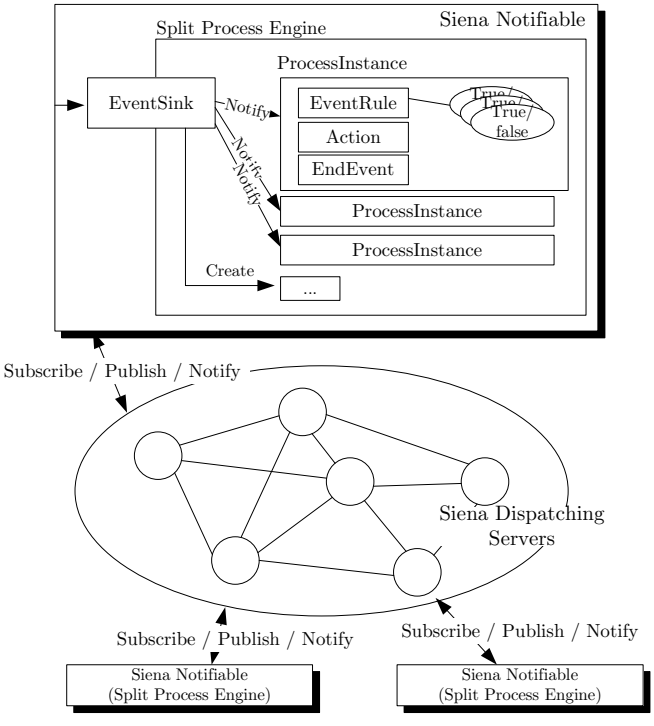


Fig. 8. Distributed Event-Based Orchestration Architecture

The working of a split process engine follows the BPMN2.0 execution semantics. Below, 3 steps are described which the process engine performs when an event notification arrives at its event sink[1] (see Fig. 8).

1. The split process engine routes the event notification to the corresponding split process instance. This is done by matching the process instance id from the event notification (found in the data payload), with the process instance ids of its already running split process instances. If no match is found, a new split process instance is started (with id equal to the process instance id situated in the event notification payload) and the notification is routed to this new split process instance.
2. In the split process instance, the event notification is matched with the correct *start event definition* in its split process description (see e.g. Fig. 6). The corresponding *event definition* will be enabled (it holds a token).
3. For every enabled *start event* in the process flow (i.e. a conjunction in the starting-rule evaluates to *true*: every *start event definition* in the *parallel multiple start event* is enabled), the rest of the split process flow is interpreted and executed. Because we adopt a really fine grained distribution of the

[1] Note that we also started the formalization of the described publish/subscribe event-based process execution [21], which we however omit here due to space limitations.

global process flow, executing the distributed process flow usually means executing only one task (e.g. invoking the *PriceCalculator* service). When the process flow reaches its end event a notification is published by the process engine to signal the end of this split process instance (i.e. the completion of the executed task). Note that the split process instance isn't deleted from the system when it reaches its end event. It is possible that there are still tokens available in some *start event definitions* of the process instance and any additional (future) event notifications can again trigger the start of the same split process instance (this is true for non-safe process models, see Fig. 4).

The published end event is routed through the event architecture, and picked up by other interested split process engines, which handle this event again with the steps described above. Eventually, the combined execution of all these split process engines have achieved the global execution of the entire, designed process flow.

6 Applicability

In this section we will focus on the major added value of our decentralized event-based orchestration, which are flexibility and adaptability. For tests on robustness and availability of distributed event based architectures (solving the single point of failure and performance bottleneck) we refer to extensive research done in the field of event based communication [11,17], as well as to other research about decentralizing the process flow [3,4,22]. The feasibility of our approach is demonstrated by our implementation of the transformation algorithm and our prototype execution architecture (see Sect. 4 and 5).

6.1 Adaptability and Change Management

The unawareness of interaction partners in an event-based communication creates a highly flexible infrastructure where components can enter and leave the architecture freely, without modifications to other components. Of course, when starting from a global process description, there is always a certain degree of dependency between the different split processes. It is designed by a process modeler that the task *Arrange Payment* should happen after the completion of task *Calculate Price*. Even with decentralized event communication, which creates a decoupling between these two tasks, the sequence dependency drawn by the process modeler still remains. Nonetheless, event communication adds some flexibility to process execution and adaptability. One advantage is a *lesser change impact when re-deploying a redesigned process flow* and another is the ability to *autonomously change starting rules of a single split process instance.*

Changing and re-deploying the global process flow. Due to the high degree of decoupling between the different split components (on execution level), re-specifying and redeploying a previously deployed process model will have lesser

Table 1. Change impact when changing the global process flow

Change Pattern	Event Orchestration	Request Orchestration
AP1-Serial Insert	2	2
AP1-Parallel Insert	2	3
AP1-Conditional Insert	2	3
AP2-Delete Process Fragment	2	3
AP3-Serial Move	3	3
AP3-Parallel Move	3	4
AP3-Conditional Move	3	4
AP4-Replace Process Fragment	2	2
AP5-Swap Process Fragment	3	3
AP8-Embed Process Frag. in Loop	2	2
AP9-Parallelize Process Frag.	n + 1	n + 2
AP10-Embed Process Frag. in Conditional Branch	2	2
AP11-Add Control Dependency	1	2
AP12-Remove Control Dep.	1	2
AP13-Update Condition	1	1
AP14-Copy Process Fragment	2	3
Total Components to Change	**32**	**41**

With n = |elements in a process fragment|

(patterns AP6-7 were left out, due to not relevant)

impact on the already running components than when using a request style of distributed orchestration. Table 1 counts the change impact according to the process change patterns introduced by Weber et al. [23]. We compared the change impact of using an event-based communication style with the change impact of using a request based distributed orchestration [4] (Fig. 2c and b). To count the change impact, we counted, for a specific change pattern, the number of split components that need to be changed, where we assume that each component has a similar weight. For example, inserting a new task between the sequential tasks *Calculate Price* and *Arrange Payment* (change patten AP1-Serial Insert), has a change impact of 2 for event orchestration: the new inserted component and the next component in the sequence (the starting rule of *Arrange Payment* needs to change). When using a request-style of communication, also 2 components need to change, the new inserted component and the component preceding the new component in the flow (*Calculate Price* needs to send a request to the new component). From table 1 it can be seen that in 9 out of 16 cases, changing the process flow with event-based execution has lesser impact on the already running infrastructure. This is a substantial benefit, because change can be costly, certainly if the components are highly distributed and not readily available for change (e.g. other people are responsible).

With proper tool support and process instance management [6], only a limited amount of components need to be redeployed and the rest can be left untouched (and running) in the architecture.

Autonomously changing a split process at runtime. Another advantage of the unawareness of interaction partners on process execution level is the autonomy of each split process. The logic on when the split process needs to start

is embedded in the split process itself. The split process has access to its own starting rules, independent of any other process engine in the global process infrastructure (unlike request based communication). This means that the entity (e.g. a person) responsible for the split process can change the starting rule of that split process independent of others. In our example, the pizza delivery boy can be responsible for his own split process holding the task *Deliver Pizza* (he has the split process running on a process engine on his PDA). Instead of waiting each time for the *Arrange Payment* `AND` *Package Order* tasks to complete, he can decide, for certain instances, to not wait for the *Arrange Payment* task and deliver the pizza nonetheless (e.g. the pizza is getting cold). Because each split process contains its own staring logic, the change is local and can be done without interfering with other split processes.

The split process engine could offer an interface to its manager, which enables creating and changing starting rules on the fly for specific process instances.

7 Related Work

In the domain of PAIS, the problem of centralized process execution is recognized by many researchers [5,22,24,3]. They all identify that, even though the actual service components are made reusable, distributed and loosely coupled through technologies like SOAP, WSDL and UDDI, the workflow- or process execution is still performed on a single central entity. Nanda et al. [24] use program dependency graphs, a tool borrowed from compiler optimization, to split up the process flow. Their goal is to reduce the network traffic involved. For the same reasons, Fdhila et al. [22] decentralize the process flow using dependency tables and Muth et al. [5] perform decentralization using state and activity charts. The eventual result is however always the same, a set of distributed control flows, where communication between the flows happens request-based. These solutions thus solve the technical issues of central orchestration (single point of failure and performance bottleneck), but still leave a tight coupled architecture, which affects robustness and adaptability.

The features of event communication are well researched in computer science [11]. Event architectures have become a standard approach to create a loosely coupled and robust communication architecture. To reap the benefits of event communication, we leverage its advantages to distributed process execution. Notice that the combination of event driven architecture and service oriented architecture is a well known topic of research [25]. The focus of this research (EDA and SOA) is however on the invocation of services (open arrowhead arrows in Fig. 2), not on the decentralization of the process flow (full arrowhead arrows in Fig. 2).

In the domain of ubiquitous [14] and agent based [26] computing, the focus is also on event communication. This focus is complementary with our approach. Events generated by ubiquitous entities (e.g. RFID sensors) or agents can be incorporated in our infrastructure, so that split process engines react to these published events directly.

The flexibility and adaptability features we advocate in this paper are in complement with the research done on *process adaptability* [6]. For example,

because of the decoupling features of event orchestration and the autonomy of split processes, the plug and play techniques of ADEPT [27] can easily be included in the architecture.

8 Conclusion and Future Work

We proposed a method that solves the issues of centralized process execution (single point of failure and performance degradation) and adds a layer of flexibility to the eventual process execution. We showed a non-intrusive transformation algorithm, that transforms a process model to smaller, event-based processes. This transformation happens at deployment time (in $O(n^2)$ time), without involvement of the process modeler. To illustrate the feasibility of the approach, we implemented the algorithm in the Atlas Transformation Language for process models defined with BPMN2.0 and developed a prototype execution architecture for the distributed event-based processes that are the result of the ATL-transformation. Each split process is run on a dedicated process engine, which differ both logically and physically from each other, and where communication between the engines is done with a publish/subscribe event-architecture. The advantage of this approach is that the decoupling features of event-based communication are adding flexibility to the process execution: there is a lesser impact on the running infrastructure when re-specifying and redeploying a process model, and the starting rules of any distributed process flow can be changed autonomic.

Future research involves working up the adaptability of distributed event-based execution and developing proper tool support to change the starting rules of split processes at run-time, as well as including *split process instance management* when changing the global process flow (which instances should be left running in the old configuration, which instances can change? [6]). We also intend to widen the scope of the transformable process elements to allow more specific process constructs (e.g. transactions) to be executed in this loosely coupled setting.

References

1. Dumas, M., Van Der Aalst, W., Ter Hofstede, A.: Process-aware information systems. Wiley Interscience, Hoboken (2005)
2. Coalition, W.M.: The workflow reference model. WfMC Documents (1995)
3. Schuler, C., Weber, R., Schuldt, H., Schek, H.J.: Scalable peer-to-peer process management-the osiris approach. In: IEEE Int. Conf. on Web Services (2004)
4. Chafle, G., Chandra, S., Mann, V., Nanda, M.: Decentralized orchestration of composite web services. In: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters, pp. 134–143 (2004)
5. Muth, P., Wodtke, D., Weissenfels, J., Dittrich, A., Weikum, G.: From centralized workflow specification to distributed workflow execution. Journal of Intelligent Information Systems 10(2), 159–184 (1998)
6. van der Aalst, W., Basten, T., Verbeek, H., Verkoulen, P., Voorhoeve, M.: Adaptive workflow: on the interplay between flexibility and support. In: Enterprise Information Systems, pp. 63–70. Kluwer Academic Publishers, Norwell (2000)

7. Recker, J.: Opportunities and constraints: the current struggle with BPMN. Business Process Management Journal 16(1), 181–201 (2010)
8. Object Management Group: Bpmn 2.0, beta 2 (June 2010), http://www.omg.org/cgi-bin/doc?dtc/10-06-04
9. Monsieur, G.: Pattern-based coordination in process-based service compositions, phd thesis. Phd Thesis, KULeuven (2010), https://lirias.kuleuven.be/bitstream/123456789/284037/1/phd_geert_monsieur.pdf
10. Hens, P., Snoeck, M., De Backer, M., Poels, G.: Decentralized Event-Based Orchestration. In: Inter. Work. on Event-Driven Business Process Management. (2010)
11. Mühl, G., Fiege, L., Pietzuch, P.: Distributed Event-Based Systems. Springer-Verlag New York, Inc., Secaucus (2006)
12. Eugster, P., Felber, P., Guerraoui, R., Kermarrec, A.: The many faces of publish/subscribe. ACM Computing Surveys (CSUR) 35(2), 131 (2003)
13. Hallerbach, A., Bauer, T., Reichert, M.: Configuration and management of process variants. In: Handbook on Business Process Management, pp. 237–255 (2010)
14. Kong, J., Jung, J.Y., Park, J.: Event-driven service coordination for business process integration in ubiquitous enterprises. Computers & Industrial Engineering 57(1), 14–26 (2009); Collaborative e-Work Networks in Industrial Engineering
15. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I., Valduriez, P.: ATL: a QVT-like transformation language. In: The 21st ACM SIGPLAN Symposium (2006)
16. Eclipse Project: Bpmn 2.0 ecore model, http://www.eclipse.org/projects/project_summary.php?projectid=modeling.mdt.bpmn2
17. Carzaniga, A., Rosenblum, D.S., Wolf, A.: Design and evaluation of a wide-area event notification service. ACM Trans. Comput. Syst. 19(3), 332–383 (2001)
18. Niblett, P., Graham, S.: Events and service-oriented architecture: the OASIS web services notification specifications. IBM Systems Journal 44(4), 869–886 (2005)
19. Geppert, A., Tombros, D.: Event-based distributed workflow execution with EVE. In: Proc. of the IFIP Int. Conf. on Distributed Systems Platforms and Open Distributed Processing, pp. 427–442 (1998)
20. Juric, M.: WSDL and BPEL extensions for Event Driven Architecture. In: Information and Software Technology (2010)
21. Hens, P., Snoeck, M., De Backer, M., Poels, G.: A Petri Net Formalization of a Publish-Subscribe Process System. In: Submitted for the 5th ACM International Conference on Distributed Event-Based Systems (DEBS) (2011)
22. Fdhila, W., Yildiz, U., Godart, C.: A flexible approach for automatic process decentralization using dependency tables. In: ICWS 2009: Proceedings of the 2009 IEEE International Conference on Web Services, pp. 847–855. IEEE Computer Society, Washington, DC, USA (2009)
23. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features-enhancing flexibility in process-aware information systems. Data & Knowledge Engineering 66(3), 438–466 (2008)
24. Nanda, M., Chandra, S., Sarkar, V.: Decentralizing execution of composite web services. ACM SIGPLAN Notices 39(10), 170–187 (2004)
25. Michelson, B.: Event-driven architecture overview. OMG report (2006)
26. Wooldridge, M.: Agent-based software engineering. IEE Proceedings Software Engineering 144(1), 26–37 (1997)
27. Dadam, P., Reichert, M., Rinderle, S., Jurisch, M., Acker, H., Goser, K., Kreher, U., Lauer, M.: Towards truly flexible and adaptive process-aware information systems. In: Information Systems and e-Business Technologies, pp. 72–83 (2008)

A State-Based Context-Aware Declarative Process Model

Pnina Soffer and Tomer Yehezkel

University of Haifa, Carmel Mountain 31905, Haifa, Israel
spnina@is.haifa.ac.il, yehezkel.tomer@gmail.com

Abstract. Declarative process models support process flexibility, which has been widely recognized as important, particularly for organizations that face frequent changes and variable stimuli from their environment. However, current declarative approaches emphasize activities and provide constraints addressing their existence and dependencies. This expressiveness is not capable of addressing the process context (namely, environment effects) and its goal. The paper proposes a declarative model which addresses activities as well as states, external events, and goals. As such, it explicitly addresses the context of a process. The model is based on the Generic Process Model (GPM), extended by a notion of activity, which includes a state change aspect and an intentional aspect. The achievement of the intention of an activity may depend on events in the environment and is hence not certain. The paper provides a formalization of the model and some conditions for verification. These are illustrated by an example from the medical domain.

Keywords: Declarative process model, Context, Generic Process Model.

1 Introduction

The importance of flexibility in process aware information systems has been widely acknowledged in the past few years. Flexibility is the ability to make changes in adaptation to a need, while keeping the essence unchanged [10]. Considering business processes, flexibility is the ability to deal with both foreseen and unforeseen changes, by varying or adapting specific parts of the business process, while retaining the essence of the parts that are not or should not be impacted by the variations [12].

Flexibility is particularly important in organizations that face frequent changes and variable stimuli from their environment. For processes that operate in a relatively stable environment, when unpredictable situations are not frequent, flexibility is not essential, as responses to all predictable situations can be defined. However, in the present business environment, where changes occur frequently and organizations have to cope with a high range of diversity, full predictability is quite rare.

Facing this reality, approaches have been proposed for enabling flexibility in business processes, as reviewed and classified in [12]. These include mechanisms of late binding and modeling, where the actual realization of a specific action is only decided at runtime as implemented in YAWL[1], and changes that can be made at runtime to a running process instance or to all instances of the process, enabled in ADEPT [11].

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 148–162, 2011.

One of the promising approaches is declarative process models (e.g., Declare [9]), which have received significant attention in recent years.

While "traditional" process models are imperative, explicitly specifying the execution order of activities through control flow constructs, a declarative process model is based on constraints, i.e., anything is possible as long as it is not explicitly forbidden. Constraint-based models, therefore, implicitly specify the execution procedure by means of constraints: any execution that does not violate constraints is possible. Using such model, the user can respond to each situation that arises, executing an activity chosen from all the ones available in compliance to the specified constraints. While allowing a high level of freedom, the approach has limitations.

First, while the human decision about which action to take is made based on the state at that specific moment, the existing models do not emphasize states. Rather, the leading concept to be modeled and monitored in the model is an activity, and constraints can be specified on the execution of a single activity or on relationships between activity executions. The process state is monitored, mainly as a trace of the activities that have been executed up to a given moment. Constraints can also relate to values of data as conditions for activity execution. However, there is no fundamental view and monitoring of state for leading process execution and decision making.

Second, to respond to changes and events that occur in the environment, these need to be addressed in the model. Generally speaking, the model should be context aware, where context is the set of inputs a process instance receives from its environment. This is particularly important when bearing in mind that flexibility is required in the first place in processes that face frequent changes in the environment.

Finally, an effective selection of action by the human operator of the process should relate to the desired outcomes to be achieved, namely, to a goal. Currently, goals are usually not an integral part of process definitions.

This paper outlines semantics for a declarative process model to overcome the three discussed limitations. To develop a consistent and complete model, we rely on the Generic Process Model (GPM) [16], which is an ontology-based theoretical process analysis framework. GPM uses states as a leading element in process representation; it has been used for analyzing the context of processes [6], and it includes goals as basic building blocks of processes. Since GPM emphasizes states and abstracts from activities in process models, in this paper it is amended to cater for activities as well.

In what follows, we start be a motivating example, demonstrating the limitations of Declare, as a representative activity-based declarative model. We then present the concepts required for our declarative model, first by informally deriving them from GPM, and then as formal definitions that set the basis for execution semantics. The use of our concepts for designing and validating processes is demonstrated through application to the running example. This is followed by discussion of related work, conclusions, and outlining of future research directions.

2 Motivating Example

This section presents a motivating example of a CT virtual cardiac catheterization process, which will be used throughout the paper as a running example. A Declare

model of the process is given in Fig. 1. The process starts with a pre CT evaluation of the patient (marked in the model as "init"). This evaluation may find the patient not fit for the scan, in which case the patient is released and the process ends. If the patient is fit and is not regularly on beta blockers, he will be administered beta blockers as preparation for the scan. Following this, either obesity (for overweight patients) or regular cardiac CT scan is performed once, based on the patient's weight. The CT scan uses a low level of radiation, serving as a first indication of arteriosclerosis. If a positive indication (evidence of calcification) is obtained, the patient is released. If the first scanning does not discover a clear evidence of calcification, a second scanning is performed. Scanning can be performed up to twice (marked in the model as "0..2" for the scan activities), and if the second scan fails, the patient is released. In case the second scan is successful, its results are deciphered and interpreted. Deciphering can be successful or unsuccessful, but in any case the patient is released. At any point in the process, some acute health situation might be identified, in which case the patient is immediately sent to an emergency room (and the process ends). In the model this is represented as ER intervention, which has an exclusive choice relation with Release patient (both end the process under different circumstances). Another possibility is that the patient may feel bad during the process (due to allergy, claustrophobic reaction, irregular heart rate, etc.). In such cases the procedure may be paused for a while and resumed after a while, when the patient feels better. Additionally, at any point in time, the patient may be released so the process ends, but unsuccessfully.

The Declare model specifies the ordering of the process activities by a precedence relation, denoting that these activities normally follow one another, but not in all cases. The activities of ER intervention and Pause examination are not mandatory in the process, and are not related to other activities by any temporal constraint.

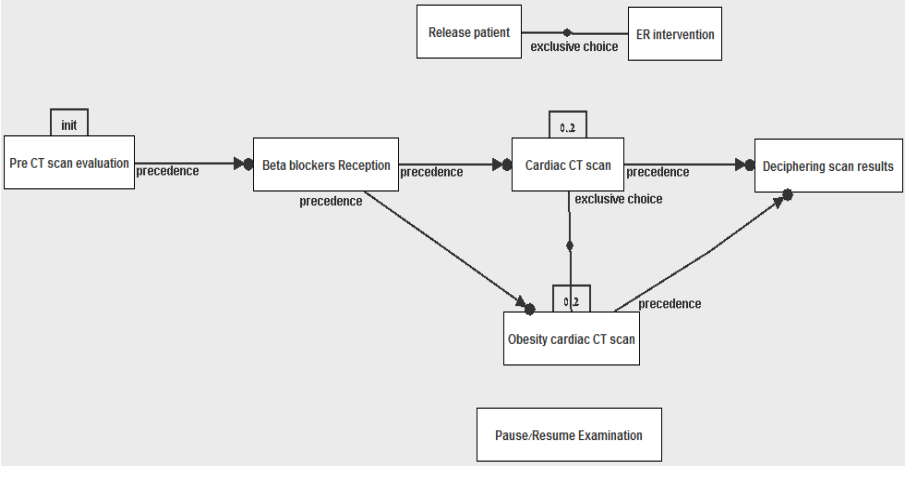


Fig. 1. The example process: a Declare model

The Declare representation supports the flexibility which is required for the process, catering for unforeseen situations and providing an immediate response based on human decision making. It also allows defining data that serves as input or

output to activities and using data as part of the activity relationship constraints. For example, the precedence between Pre CT evaluation and beta blockers reception is conditioned by a "fit" value of the data item Candidacy, assigned by the Pre CT evaluation activity. Note that the process could also be specified using an imperative modeling language (e.g., BPMN). However, this would require a very complex model to specify all the possible variations.

Despite all the discussed advantages, we claim that this representation is not expressive enough, and it leaves parts of the flow logic of the process to human judgment, where this logic is clear and needs to be specified and enforced. Examples include: (1) a second scan is performed only if a clear evidence of calcification has not been obtained in the first scan; (2) Beta blockers reception is needed only for patients who do not use them regularly; (3) Pause examination and ER intervention are performed when the patient has some irregularity or when an acute problem is identified, respectively. These are constraints on the process flow, which cannot be expressed as relationships between activities or existence constraints on the activities. Furthermore, the model does not specify conditions under which the process terminates. Implicitly, the process cannot end before all the existence constraints on its activities are satisfied. However, in our example the only mandatory activity is Pre CT evaluation, while termination of the process is possible under defined conditions. It is possible to add a set of negation constraints, negating any activity after the activities of Release patient or ER intervention are performed. This, however, would result in a loaded model which is hard to follow.

Roughly speaking, we may conclude that Declare does not support constraints that relate to the context of the process and to its goal, where context refers to all environmental effects on a specific execution of the process. These may be general environment conditions (see [20]) or specific case properties [6]. In Declare they are assumed to be addressed by human judgment when the process is executed, enabled by the flexibility of the specification.

In the following sections we present an approach derived from theory, which enables a process specification that captures contextual constraints and process goals, while supporting flexibility. The theoretical basis provides for a complete set of constructs, capable of fully expressing the business logic of processes.

3 Ontological State-Based View

The starting point of our discussion is the Generic Process Model [15][16], which is a process analysis framework, building on Bunge's ontology [5]. GPM emphasizes states, events, and goals, which, as shown above, are not well addressed in current declarative process models.

The focus of attention in GPM is the *domain* where the process takes place. The process domain is a *composite thing*, represented by a set of *state variables*, whose values at a moment in time denote the *state* of the domain. A state can be *unstable*, in which case it will transform according to the *transition law* of the domain (*internal event*), or *stable*, namely, it will not change unless invoked by an event in the environment (*external event*). GPM views an enacted process as a set of state transitions in the process domain. Transitions result either from *transformations* within the

domain (reflecting its transition law), or from actions of the environment on the domain. A process ends when the domain reaches a desired (*goal*) state, which is stable and where no more changes can occur due to domain dynamics.

A process model is an abstract representation of the process, defined as follows.

Definition 1 *(GPM process model): A process model in a given domain is a tuple <I, G, L, E>, where*

I: the set of possible initial states – a subset of unstable states of the domain.
G: the goal set – a subset of the stable states reflecting stakeholders' objectives.
L: the transition law defined on the domain – specifies possible state transitions as mappings between sets of states.
E: a set of relevant external events that can or need to occur during the process.

As noted, the focus of attention in GPM is the process domain. The domain sets the boundaries of what is fully controlled by the process and its operators, and what is not. This distinction enables us to define the context of a process [6] as the set of environmental effects on the process, which are twofold. First, the properties of the specific case handled by a process instance – these are assumed to exist at the initiation of the case, although not all their values are necessarily known at that point in time. Second, actions of the environment during process execution – these are manifested as external events. External events are events (state transitions) in the environment of the process domain, which affect the state of the domain through mutual state variables. Taking place outside the domain, they are not controlled by it. The occurrence of an external event can be unanticipated, but even if we anticipate the occurrence, the exact time when it would take place and its resulting state are usually not predictable. In particular, it is different for every process instance. Hence, the E and I elements in a process model represent contextual elements.

A second advantage of GPM is that it explicitly addresses the goal of a process, enabling the design of a process to achieve its goal, and assessing the validity of a process design against its defined goal. At runtime, achieving a goal state marks the termination of a process instance.

However, the transition law of GPM, which is a mapping between sets of states, is an abstract notion. Specifically, as indicated in Definition 1, GPM's process model abstracts from activities, which are how state changes are brought about. Hence, to make GPM an appropriate basis for declarative process models, the law needs to be decomposed into activities and constraints. To do so, a clear understanding of what an activity is needs to be developed.

Activities are the means for achieving internal events. Since internal events usually affect a subset of the domain state variables, namely, a sub-domain, and since different internal events can occur concurrently in independent sub-domains [14], we may address an activity as an internal event in a sub-domain. However, a sub-domain may change its state through a series of internal events in an almost continuous manner. What makes a specific trajectory be considered an activity is the intention that drives it. For example, consider the activity of Pre CT evaluation, which entails actions such as measuring the patient's blood pressure and heart rate, performing an electrocardiogram, and others. We consider all these actions as parts of one activity, distinguished by the one aim to be achieved. Intentions can be of achieving, maintaining, or avoiding a state [7].

We define an activity as an internal event in a sub-domain, intended to achieve a defined change in its state.

According to our model the state change brought by an activity is deterministic. However, the state variables whose values are changed might be mutual state variables of the process domain and its environment. In such cases, the environment is affected and its state might become unstable. This, in turn, causes transformations (events) in the environment, and these events might, again, affect the process domain. Thus, an activity that acts on the environment might lead to an external event in response. Since external events are not controlled by the process domain and their outcome is unpredictable, this might seem as if the outcome of the activity is unpredictable (especially if the reaction is immediate). Nevertheless, we specify only the controllable change within the process domain as part of the activity, and distinguish the uncontrollable change as an external event invoked by the activity and its effect on the process environment. For example, consider a basketball player throwing the ball to the basket. The activity has ended once the ball is in the air, which is an unstable state of the environment. The movement of the ball in the air is not controlled by the player. The resulting event can be that the ball has missed or hit the basket, and it is an external event, not completely predictable. The actual value resulting from the external event will be determined at runtime. Note that in this example, the intention of the activity was to bring about a state where the ball is in the basket, but this can only be achieved by an external event, and with uncertainty.

Activities can hence be classified to two classes: (a) activities that affect only state variables which are intrinsic to the process domain. Such activities cause a fully predictable change that achieves the intention associated with the activity. (b) Activities that affect the state of the environment and invoke an external event. For these activities the specified (and predictable) change in the state does not necessarily correspond to the intended change. It may not even relate to the same state variables.

Note that activities of class (a), namely activities that operate on intrinsic domain state variables, may also entail changes in state variable values that depend on input given by the user at runtime. As an example, consider a process where the price of a product is determined. The activity of pricing might require the user to set a price based on his individual judgment of the appropriate profit margin. This will be manifested as user input at runtime, but is considered part of the activity since the value is controlled within the process domain.

We now consider constraints. The GPM law can be represented by three types of constraints: (a) initiation constraints, setting the possible initial set of states, (b) transformation constraints that specify the relationship state-activity, namely, states that are preconditions for activities, (c) Termination (goal) constraints, defining the set of states where the process can terminate having achieved its goal. In addition, we can define a fourth type of constraint – environment response constraints that place external events as response to activities that affect the environment. Note that the actual effect of these events on the domain is not known, nor is the exact time of their occurrence. Below we discuss each of these four constraint types.

Initiation constraints – determine values of state variables to specify the conditions under which the process can begin (e.g., when a patient arrives at the clinic). In particular, all the state variables that count the occurrences of activities are set to zero. Note that the initiation constraints do not determine the exact state on which a process

instance begins. This exact state also includes values of contextual properties which characterize each specific case (e.g., the weight of a patient).

Transformation constraints – include two kinds of constraints: enabling constraints and triggering constraints. Enabling constraints relate activities to the sets of states when they can be activated. Note that the state that follows the execution of an activity is directly calculated from the state that precedes it and the change it causes.

Triggering constraints specify sets of states when an activity must be activated, so when a state in this set is reached the activity will immediately fire.

Termination constraints – determine sets of states where the process terminates. There might be two kinds of termination states. First, goal states, which are stable states the process is intended to achieve. Once a state in the goal set is reached, the process terminates. Second, exception states, which are stable states where the process terminates without achieving what it is intended to achieve. For example, the virtual cardiac catheterization process can terminate when it is found out the patient is not fit for scan or when an ER intervention is needed. These are exception states. Note that the process may include stable states which are not defined as termination states. If such a state is reached, the process waits for an external event to reactivate it. In the virtual cardiac catheterization process a state after the examination has been paused is stable, waiting for an external event when the patient feels better and has no irregularity to resume the process.

Environment response constraints – relate external events to activities that invoke them. Note that external events can also occur unexpectedly. In many cases the external event does not necessarily immediately follow the activity; there might be some time elapse between them. Hence, the relationship is of precedence.

Finally, it can be shown that the combination of initiation, termination, triggering, and enabling constraints is sufficient for expressing all the constraint types available in Declare. Our set of constraints provides these operations with respect to a broader scope, including context and goal. Hence, it provides a richer expressive power.

4 Formalization

Following the above discussion, we now formalize the proposed constructs and execution semantics.

Definition 2 *(process model): Let D be a domain represented by its state variables vector* $X=(x_1,x_2,\ldots x_n)$*. Let* v_i *be the domain of values of state variable* x_i*,* $V=(v_1,v_2,\ldots v_n)$*. A process model M over D is a tuple (I, G, A, Const, E), where*

I: a set of states satisfying the initiation constraints
G: a set of states satisfying termination constraints; $G=Gg\cup Ge$*; Gg includes states defined as the goal of the process, Ge are states of exceptional termination.*
A: a set of activities
Const: a set of constraints
E: a set of external events.

In general, sets of states are specified by predicates over the state variable vector. Hence, given predicates C_I, C_{Gg}, and C_{Ge} that specify initiation, goal, and exceptional termination conditions respectively, we obtain:

$I=\{s \mid C_I(X)=TRUE\}$; $Gg=\{s \mid C_{Gg}(X)=TRUE\}$; $Ge=\{s \mid C_{Ge}(X)=TRUE\}$.

As discussed in the previous section, activities are intentional changes in the state of a sub-domain. Following this, the specification of an activity includes two elements: the change (delta) it brings about to the state of the sub-domain and the intended set of states to be achieved.

Definition 3 *(activitiy): Let $\delta(X)$ be a function, $\delta: V \rightarrow V$. Then $a \in A$: $(\delta(X), \gamma(X))$, where γ is a predicate denoting the set of states intended to be achieved by the activity.*

Note that δ usually implies a change in a subset of the domain state variables, which are the ones affected by the activity. In particular, a state variable counting the number of executions of the activity will be raised by 1. Also note that if $\gamma(X)$ includes negation operators, then the intention of the activity is to avoid a set of states. As well, if $\gamma(X)$ refers back to the set of states that precede the activity (except for the state variable that counts the executions of the activity), then the intention of the activity is to maintain an existing state.

The set of constraints includes the transformation and environment response constraints, since initiation and termination constraints are specified in *I* and *G*. As discussed in the previous section, transformation constraints include enabling constraints and triggering constraints.

Definition 4 *(enabling constraint): Let $a \in A$, θ_a a predicate, $En(a)=\{s \mid \theta_a(X)=TRUE\}$, then a can fire for every $s \in En(a)$.*

Definition 5 *(triggering constraint): Let $a \in A$, τ_a a predicate, $Tr(a)=\{s \mid \tau_a(X)=TRUE\}$, then a must fire for every $s \in Tr(a)$.*

In order to define the environment response constrains, we first need to define the external events element in the model. In a process model an external event is an occurrence we make no a-priori assumptions about (e.g., regarding its effect on the state of the domain). However, some external events which are expected to occur are expected to affect a subset of the domain state variables and assign them some value within its domain of possible values. The actual state that follows an external event will become known at runtime as input made by the user.

Definition 6 *(external event): An external event $e \in E$: $\{(x_i, v_i) \mid x_i \in X, v_i \in V\}$*

In words, an event is defined by a subset of the domain state variables which it affects, resulting in values within their domain of values.

Environment response constraints relate expected external events to the activity that invokes them.

Definition 7 *(environment response constraint): Let $a \in A$, $e \in E$. An environment response constraint Er:(a,e) denotes that e always occurs eventually after a.*

Since the occurrence of external events is not within the process control, environment response constraints cannot be enforced at runtime. Nevertheless, they are specified in

the model so they can be considered at process design time, and can be taken into account when planning ahead in runtime.

Another possibility of external event would be some general unforeseen change in the environment. It would be modeled as an "empty" event without any prior assumption, not related to any of the defined constraints. Examples include power or hardware failure in one of the CT system main components: the X-ray Tube, collimators, detectors or data acquisition system (DAS).

Based on the above definitions, we now provide a semi-formal description of the execution mechanism of a process. When a process is executed the state s at a moment in time is the values of x_i at that moment. When a process instance is created, the initial state is $s_I \in I$, satisfying the initiation constraints and including state variable values that represent contextual properties (initiated by user input). After initiation, the state at every moment is considered. For a given state s, the set of enabled activities is $A_{En}=\{a \mid s \in En(a)\}$; the set of triggered activities is $A_{Tr}=\{a \mid s \in Tr(a)\}$; an activity in A_{En} can be executed; an activity in A_{Tr} must be executed at that moment.

State changes can occur due to activity completion or to external events. Assume an activity a starts when the state is s. Then the state on completion of a is $\delta_a(s)$. The occurrence of event e requires the user to provide specific values for each state variable affected by the event; these values set the state that follows the event. Termination of the process is also determined based on the state, so if $s \in G$ then the process terminates.

Note that the intention component of the activity specification does not take part in the execution. However, it plays an important role at process design, as detailed in the next section. In addition, the intention is meaningful for planning ahead at runtime.

5 Specifying a Process

This section demonstrates how the proposed semantics can be used for expressing the running example of the virtual cardiac catheterization process.

We start by defining the state variables of the domain and their possible range of values (Table 1). Table 1 also provides the initial value of each state variable, defining the initial set of states of the process, I.

Note that initial values are set for a subset of the state variables, while state variables whose initial value is not specified stand for contextual properties. These need to be initialized to represent specific case properties. In our example process the relevant contextual properties are overweight of the patient and whether the patient is regularly on beta blockers. Also note a set of state variables that count the executions of each activity, as seen in their possible values – natural numbers from 0 to infinity.

The termination set is comprised of two sets of states, Gg of desired (goal) states and Ge of undesired termination states. Considering our example:

Gg={s | (Scan Deciphering = "successful") $\wedge$ (Patient Released="Yes")}

Ge={s | ((Deciphering scan results $\neq$ "successful") $\wedge$ (Patient Released="Yes")) $\vee$ (ER intervention="Yes")}

Ge stands for two possible cases of termination – when the patient is released without having reached successfully deciphered images (e.g., not found fit to scanning, or after calcification has been discovered), or when ER intervention is needed.

Table 1. State variables in the example process and their initial values

State variable	Values	Initial value	State variable	Values	Initial value
Compatibility	{Null, Fit, Not fit}	Null	Beta Blockers	{Given, Not given}	
Over-weight	{Yes, No}		Calcification	{Found, Not found}	Not found
Images	{Null, Successful, Unsuccessful}	Null	Deciphering results	{Null, Successful, Unsuccessful}	Null
Acute problem	{Undiscovered, Discovered}	Undiscovered	Irregularity	{Null, Appear, Disappear}	Null
Patient released	{Yes, No}	No	ER Intervention	{Yes, No}	No
Pre CT Evaluation	$[0, \infty)$	0	Beta blockers reception	$[0, \infty)$	0
Cardiac CT scan	$[0, \infty)$	0	Obesity CT scan	$[0, \infty)$	0
Deciphering scan results	$[0, \infty)$	0	Pause/resume examination	{0, 1}	0

The activities of the process are specified in Table 2 in terms of the function δ, relating to specific state variables, and the predicate γ. The table also specifies for every activity a the predicates θ_a and τ_a that define the related transformation constraints *En(a)* and *Tr(a)*, respectively.

To illustrate the specification of activities and their related transformation constraints, let us consider the activity Obesity CT scan, whose δ relates to the execution counter of the activity, raising it by 1. Recall, this activity can be performed up to twice. Ideally, after two executions a state will be reached where γ is achieved, namely (Calcification ="Not Found") ^ (Images = "Successful"). The enabling set of this activity is when (Compatibility="Fit") $\wedge$ (Beta blockers = "Given") $\wedge$ (Over-weight = "Yes") $\wedge$ (Calcification = "Not Found") $\wedge$ (Cardiac CT scan = 0) $\wedge$ (Paused/Resume Examination = 0) $\wedge$ (Obesity CT scan < 2), denoting that (a) the activity can start after beta blockers are given (either in the process or in its context) and compatibility is evaluated and found fit (this condition is needed in case beta blockers are given contextually), (b) the activity is executed only for patients with over-weight (in which case Cardiac CT scan cannot be performed), (c) the activity can only be performed twice, and it is not repeated if calcification is found, and (d) the activity cannot start when the examination is paused.

As another example, consider the activity Pause/resume examination, whose role is to pause the examination when the patient has irregularities, and to resume it when the irregularity disappears. The activity can be triggered when irregularity appears if the

examination is not already paused ((Irregularity = Appear) ∧ (Paused/Resume Examination = 0)). Then the activity stops the examination (see Paused/Resume Examination → 1) if (Paused/Resume Examination = 0) in the δ column). Alternatively, the activity is triggered when the examination is already paused and the irregularity disappears, in which case the activity resumes the examination.

Table 2. Activities and corresponding transformation constraints

Activity	**δ**	**γ**	**Transformation constraints**
Pre CT Evaluation	Pre CT Evaluation→ Pre CT Evaluation + 1	Candidacy= "fit"	θ_a: (Candidacy="null")∧ (Paused/Resume Examination=0)
Beta blockers Reception	(Beta blockers Reception→Beta blockers Reception +1) ∧ (Beta blockers = "Given")	Beta blockers = "Given"	θ_a:(Compatibility="Fit") ∧ (Beta blockers≠"Given")∧ (Paused/resume Examination = 0)
Cardiac CT scan	Cardiac CT scan) → Cardiac CT scan +1	(Calcification ="Not Found") ^ (Images = "Successful")	θ_a: (Compatibility="Fit") ∧ (Beta blockers = "Given") ∧ (Over-weight = "No") ∧ (Calcification = "Not Found") ∧ (Obesity CT scan = 0) ∧ (Paused/Resume Examination = 0) ∧ (Cardiac CT scan < 2)
Obesity CT scan	Obesity CT scan) → Obesity CT scan +1	(Calcification ="Not Found") ^ (Images = "Successful")	θ_a: (Compatibility="Fit") ∧ (Beta blockers = "Given") ∧ (Over-weight = "Yes") ∧ (Calcification = "Not Found") ∧ (Cardiac CT scan = 0) ∧ (Paused/Resume Examination = 0) ∧ (Obesity CT scan < 2)
Deciphering scan results	Deciphering scan results→ Deciphering scan results + 1	Deciphering results= "Successful"	θ_a: Images = "Successful"
ER Intervention	ER intervention = "Yes"	ER intervention = "Yes"	θ_a: Patient released = "No" τ_a: Acute problem= "Discovered"
Release Patient	Patient Released = "Yes"	Patient Released = "Yes"	θ_a: (ER intervention="No")
Pause / resume examination	Paused/Resume Examination → 1) if (Paused/Resume Examination = 0); (Paused/Resume Examination → 0) if (Paused/Resume Examination = 1)	(Paused/Resume Examination=1∧ Irregularity= Appear) ∨ (Paused/Resume Examination=0∧ Irregularity= disappear)	τ_a: ((Irregularity = Appear) ∧ (Paused/Resume Examination = 0)) ∨ ((Irregularity = Disappear) ∧ (Paused/Resume Examination = 1))

Note that there are activities such as Beta blockers reception, where for a state preceding the activity $s \in En(a) \cup Tr(a)$, the change achieved with certainty $\delta_a(s)$ satisfies the intention γ_a. These are activities that achieve their intention with certainty, not depending on external events. For other activities (e.g., Pre CT Evaluation), an external event is expected in response to the activity, for a state satisfying γ_a to be

achieved. As previously discussed, in these cases the intention of the activity may not be achieved, depending on the values set by the external event.

To complete the process specification, we define the set of external events *E*, which, together with the uninitiated variables in Table 1, form the context of the process. Table 3 includes external events and their associated environment response constraints (column "Response to" in the table). Some of the external events are expected in response to specific activities, while some can occur unexpectedly, in which case the "Response to" column is blank.

Table 3. External events in the example process

External event	Affected state variables	Response to
Candidate compatibility	Candidacy	Pre CT scan evaluation
Calcification discovery	Calcification	Cardiac CT scan/ Obesity cardiac CT scan
Image generation	Images	Cardiac CT scan/ Obesity cardiac CT scan
Deciphering outcome	Deciphering results	Deciphering scan results
Acute health problem	Acute problem	
Irregularity appearance	Irregularity	

To illustrate, consider the event Calcification discovery. This event is expected in response to a CT scan (either obesity or regular). It may change the value of the state variable Calcification from Not Found to Found (see Table 1).

Having specified the process, we now present four conditions which are necessary for the specification to be valid, namely, for the process to achieve its goal.

Condition 1 ***(concurrency):*** *For every a, a'∈A, if* $Tr(a) \cap TR(a') \neq \emptyset$ *then* δ_a *and* $\delta_{a'}$ *do not affect the same state variables.*

This condition is intended to ensure that two activities that are triggered by the same state (namely, must be performed concurrently), do not change the values of shared state variables. For a discussion of this condition, see [14]. Our process includes two activities with triggering conditions: Pause/resume examination and ER intervention. Their triggering constraints are not overlapping, hence Condition 1 is not breeched.

Condition 2 ***(sequence of intended states):*** *There exists at least one sequence of states* $(s_1, s_2, \ldots s_n)$ *such that* $s_1 \in I$, $s_n \in Gg$, *and let* $s_i \in \{s | \gamma_{ai} = TRUE\}$ *then for* $i=2 \ldots n$ $s_{i-1} \in En(a_i) \cup Tr(a_i)$.

This condition requires the existence of at least one sequence of states leading from an initial state to the goal of the process. Note that the sequence is established when the enabling or triggering set of each activity is in the intended set of the previous one. This means that the activity is enabled either immediately and certainly after an activity whose intention is achieved by its δ, or after an external event responding to the previous activity has achieved its intention.

In our example it can be noticed that for, e.g., the sequence of activities Pre CT scan evaluation, Beta blockers reception, Cardiac CT scan, Deciphering scan results, and Release patient, the enabling set of each activity satisfies the intention of the previous one. As well, the first activity (Pre CT scan evaluation) is enabled at I and the last one can lead to a state in Gg.

Also note that there might be other sequences which do not lead to the goal set. These, however, should end on a termination state in Ge and not on any other state. In other words, continuation of the process should be enabled for any state which is not in G. To ensure this continuation, we require the following two conditions.

Condition 3 ***(process continuation - activities):*** *Let $a \in A$ such that γ_a is not achieved by δ_a. Then $\exists$ an external event $e \in E$ and an environment response constraint Er:(a,e).*

For example, δ of Deciphering scan results increases the execution counter of this activity by 1, while its intention is to reach a state where Deciphering results are Successful. This can be achieved by the external event Deciphering outcome, which is related to the activity by an environment response constraint.

Condition 4 ***(process continuation - events):*** *For every external event e: $\{(x_i, v_i)\}$, for every value v_i that state variable x_i can assume, the resulting state s satisfies (1) $s \in G$ or (2) $\exists$ activity $a \in A$ such that $s \in En(a) \cup Tr(a)$.*

This condition requires that every external event either leads to a state in the termination set or to a state where at least one activity is enabled / triggered. For example, the event Acute health problem leads to a state where *Acute problem = "Discovered"*. This state triggers the activity of ER intervention.

6 Related Work

Substantial research efforts have been invested in declarative process models in recent years. Most notably, Declare [9], a modeling and execution platform, which also allows changing the model at runtime and performing some verification. Methodological issues that concern declarative process models have also been investigated. Examples include life-cycle support [18], user assistance [12], and usability evaluation [17]. Life-cycle support [18] is said to ensure better understandability and maintainability of declarative processes over the process life-cycle, based on the ideas of Test Driven Development [4] and Automated Acceptance. User assistance includes recommendations which are generated based on similar past process executions by considering specific business objectives [13]. An initial usability evaluation using the Alaska simulator, has indicated that humans are capable of coping with flexibility and can effectively plan in an agile manner [17].

Declarative model segments have also been used for managing imperative models. Ly et. al. [8] developed a framework for integrating constraints into adaptive process management systems in order to ensure semantic correctness of running processes at any time. Awad et al. [3] suggested a technique to accomplish the verification of process models against imposed compliance rules by using BPMN-Q queries.

All these approaches basically employ an activity-based view, with Linear temporal logic (LTL)-based constraints. These constraints are capable of defining rules on the existence of activities and on dependencies among them. Goals are usually not

specified or addressed, although some goal consideration is included in the Alaska simulator [17]. The tool uses a journey as metaphor for businesses process and determines typical goal as the overall business value of the journey, i.e., minimization of cost, cycle time or the optimization of quality or customer satisfaction. Goals of this kind are not addressed in this paper (in GPM terminology they are called soft-goals [15]). Rather, we address "hard" goals which mark the termination of the process. Soft-goals mainly affect planning and will be addressed as future research.

A different and early approach is presented by [2], who defined a business process pattern based on the state-oriented approach that includes a state space, a goal, and valid movements in the state space. Constraints are not made formally and explicitly, but roughly in the form of valid movements. The movements refer only to the changes in the constructed state space and abstract from activities.

In summary, as compared to the existing declarative process modeling approaches, the approach presented in this paper has an extended expressiveness, enabling all LTL-based constraints and adding state information. Furthermore, it supports contextual constraints, which are not possible in existing approaches.

7 Conclusions

Declarative process models support flexibility in process aware information systems. However, current declarative models are mainly activity-based, relying on Linear Temporal Logic for the constraints they entail. As a result, the business logic related to the context of the process and to its goal is basically applied by human and not supported by the model.

The model proposed in this paper constrains the execution of activities based on state, which reflects activity execution as well as case properties and results of events in the environment of the process. As such, it is context aware and suitable for highly diverse and frequently changing environments, where process flexibility is particularly important. Furthermore, an explicit goal specification can guide execution towards this goal and serve for validating the process at design time.

A theoretical contribution of the paper is the activity definition, which makes a clear distinction between the certain change brought about by it and the intended change, which may or may not depend on environment response invoked by the activity. The intentional aspect of an activity is shown to be of importance for designing the process and for planning ahead for reaching the goal. Yet, it can be ignored by an execution engine for simplicity, as it has no role in the actual execution mechanism.

The paper demonstrates the specification of an example process using the proposed model. This specification, however, is not graphical. An appropriate graphical representation to increase the usability of the model by humans is still needed. We intend to consider the adaptation of the graphical notation used by Declare to the additional expressiveness required by our model as future work. The paper also provides four conditions a process specification needs to meet to be valid. As noted, these are necessary conditions for the validity of the process, but not necessarily sufficient. Full validation and verification of a process specification is also planned as future research, as well as the utilization of AI planning algorithms to support goal achievement from a given state. Defining a comprehensive and complete event library may also be addressed in future work.

References

1. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: Yet Another Workflow Language. Information Systems 30(4), 245–275 (2005)
2. Andersson, B., Bider, I., Johannesson, P., Perjons, E.: Towards a Formal Definition of Goal-Oriented Business Process Patterns. Business Process Management Journal 11(6) (2005)
3. Awad, A., Decker, G., Weske, M.: Efficient compliance checking using BPMN-Q and temporal logic. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 326–341. Springer, Heidelberg (2008)
4. Beck, K.: Test Driven Development: By Example. Addison-Wesley, Reading (2002)
5. Bunge, M.: Treatise on Basic Philosophy. Ontology I: The Furniture of the World, vol. 3. Reidel, Boston (1977)
6. Ghattas, J., Soffer, P., Peleg, M.: A formal model for process context learning. In: Proc. BPI 2009, Ulm, Germany (2009)
7. Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Guided Tour. In: 5th Int'l Symp. on RE, pp. 249–261. IEEE CS Press, Los Alamitos (2001)
8. Ly, L.T., Rinderle, S., Dadam, P.: Integration and veri_cation of semantic constraints in adaptive process management systems. Data and Knowledge Engineering 64, 3–23 (2008)
9. Pesic, M., Schonenberg, M.H., Sidorova, N., van der Aalst, W.M.P.: Constraint-based workflow models: Change made easy. In: Chung, S. (ed.) OTM 2007, Part I. LNCS, vol. 4803, pp. 77–94. Springer, Heidelberg (2007)
10. Regev, G., Bider, I., Wegmann, A.: Defining business process flexibility with the help of invariants. Software Process Improvement and Practice 12, 65–79 (2007)
11. Reichert, M., Rinderle, S., Dadam, P.: ADEPT workflow management system: In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 370–379. Springer, Heidelberg (2003)
12. Schonenberg, H., Mans, R., Russell, N., Mulyar, N., van der Aalst, W.M.P.: Process Flexibility: A Survey of Contemporary Approaches. In: Dietz, et al (eds.) CIAO! And EOMAS 2008. LNBIP, vol. 10, pp. 16–30. Springer, Berlin (2008)
13. Schonenberg, H., Weber, B., van Dongen, B.F., van der Aalst, W.M.P.: Supporting flexible processes through recommendations based on history. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 51–66. Springer, Heidelberg (2008)
14. Soffer, P., Kaner, M., Wand, Y.: Assigning Ontology-Based Semantics to Workflow nets". Journal of Database Management 21(3), 1–35 (2010)
15. Soffer, P., Wand, Y.: On the Notion of Soft Goals in Business Process Modeling. Business Process Management Journal 11(6), 663–679 (2005)
16. Soffer, P., Wand, Y.: Goal-driven multi-process analysis. Journal of the Association of Information Systems 8(3), 175–203 (2007)
17. Weber, B., Pinggera, J., Zugal, S., Wild, W.: Handling events during business process execution: An empirical test
18. Weber, B., Reijers, H.A., Zugal, S., Wild, W.: The declarative approach to business process execution: An empirical test. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 470–485. Springer, Heidelberg (2009)
19. Zugal, S., Pinggera, J., Weber, B.: Toward Enhanced Life-Cycle Support for Declarative Processes. To appear in IJISMD
20. Ploesser, K., Janiesch, C., Recker, J., Rosemann, M.: Context Change Archetypes:Understanding the Impact of Context Change on Business Processes (2009)

The Impact of Testcases on the Maintainability of Declarative Process Models

Stefan Zugal, Jakob Pinggera, and Barbara Weber

University of Innsbruck, Austria
{stefan.zugal,jakob.pinggera,barbara.weber}@uibk.ac.at

Abstract. Declarative approaches to process modeling are regarded well suited for highly volatile environments as they provide a high degree of flexibility. However, problems in understanding and maintaining declarative process models impede their usage. To compensate for these shortcomings Test Driven Modeling has been proposed. This paper reports from a controlled experiment evaluating the impact of Test Driven Modeling, in particular the adoption of testcases, on process model maintenance. Thereby, students modified declarative process models, one model with the support of testcases and one model without the support of testcases. Data gathered in this experiment shows that the adoption of testcases significantly lowers cognitive load and increases perceived quality of changes. In addition, modelers who had testcases at hand performed significantly more change operations, while at the same time the quality of process models did not decrease.

Keywords: Declarative Business Process Models, Test Driven Modeling, Empirical Research.

1 Introduction

In today's dynamic business environment, the economic success of an enterprise depends on its ability to react to various changes like shifts in customer's attitudes or the introduction of new regulations and exceptional circumstances [1], [2]. Process-Aware Information Systems (PAISs) offer a promising perspective on shaping this capability, resulting in growing interest to align information systems in a process-oriented way [3], [4]. Yet, a critical success factor in applying PAISs is the possibility of flexibly dealing with process changes [1]. To address the need for flexible PAISs, competing paradigms enabling process changes and process flexibility have been developed, e.g., adaptive processes [5], [6], case handling [7], declarative processes [8], data driven processes [9] and late binding and modeling [10] (for an overview see [11]).

Especially declarative processes have recently attracted the interest of researchers, as they promise to provide a high degree of flexibility [11]. Although the benefits of declarative approaches seem rather evident [8], they are not widely adopted in practice yet. In particular, as pointed out in [12], [13], [14], understandability problems and maintainability problems hamper the usage of declarative process models. An approach tackling these problems, the so-called Test

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 163–177, 2011.

Driven Modeling (TDM) methodology, is presented in [14]. TDM aims at improving the understandability and maintainability of declarative process models as well as the communication between domain expert and model builder by adopting the concept of *testcases* from software engineering. While the proposed concepts seem to be beneficial from a theoretical point of view, no empirical evaluation has been conducted yet. The goal of this paper is to pick up this need and to investigate empirically, whether the adoption of TDM—in particular the usage of testcases—has the intended positive effects on the *maintainability* of declarative process models.

To this end, we performed a controlled experiment at the University of Innsbruck, letting its participants conduct changes to declarative process models with and without test support. This paper reports on the experiment and its results, starting with necessary background information and prerequisites in Section 2. Then, Section 3 describes the experimental setup, whereas Section 4 deals with the actual experimental execution, data analysis and discussion. Related work is presented in Section 5 and, finally, Section 6 concludes the paper with a summary and an outlook.

2 Background

This section provides background information needed for the further understanding of the paper. Section 2.1 introduces declarative processes, while Section 2.2 discusses associated problems. Then Section 2.3 sketches how TDM aims at resolving these problems. Afterwards, Section 2.4 introduces Test Driven Modeling Suite that implements the concepts of TDM and was used as experimental platform.

2.1 Declarative Processes

There has been a long tradition of modeling business processes in an imperative way. Process modeling languages supporting this paradigm, like BPMN, EPC and UML Activity Diagrams, are widely used. Recently, *declarative approaches* have received increasing interest and suggest a fundamentally different way of describing business processes [12]. While imperative models specify exactly *how* things have to be done, declarative approaches only focus on the logic that governs the interplay of actions in the process by describing the *activities* that can be performed, as well as *constraints* prohibiting undesired behavior. An example of a constraint in an aviation process would be that crew duty times cannot exceed a predefined threshold. Constraints described in literature can be classified as execution and termination constraints. *Execution* constraints, on the one hand, restrict the execution of activities, e.g., an activity can be executed at most once. *Termination* constraints, on the other hand, affect the termination of process instances and specify when process termination is possible. For instance, an activity must be executed at least once before the process can be terminated. Most constraints focus either on execution *or* termination semantics, however,

some constraints also combine execution and termination semantics (e.g., the succession constraint [12]).

To illustrate the concept of declarative processes, a declarative process model is shown in Fig. 1 a). It contains activities *A* to *F* as well as constraints *C1* and *C2*. *C1* prescribes that A must be executed at least once (i.e., *C1* restricts the termination of process instances), whereas *C2* specifies that *E* can only be executed if *C* has been executed at some point in time before (i.e., *C2* imposes restrictions on the execution of activity E). In Fig. 1 b) an example of a process instance illustrates the semantics of the described constraints. After process instantiation, *A*, *B*, *C*, *D* and *F* can be executed. *E*, however, cannot be executed as *C2* specifies that *C* must have been executed before (cf. grey bar in Fig. 1 b) below "E"). Furthermore, the process instance cannot be terminated as *C1* is not satisfied, i.e., *A* has not been executed at least once (cf. grey area in Fig. 1 b) below "Termination"). The subsequent execution of B does not cause any changes as it is not involved in any constraint. However, after *A* is executed, *C1* is satisfied, i.e., *A* has been executed at least once and thus the process instance can be terminated (cf. Fig. 1 b)—after *e4* the box below "Termination" is white). Then, *C* is executed, satisfying *C2* and consequently allowing *E* to be executed (the box below "E" is white after *e6* occurred). Finally, the execution of *E* does not affect any constraint, thus no changes with respect to constraint satisfaction can be observed. As all termination constraints are still satisfied, the process instance can still be terminated.

As illustrated in Fig. 1, a process instance can be specified through a list of *events* that describe changes in the life-cycle of *activity instances*, e.g., *"e1: B started"*. In the following, we will denote this list as *execution trace*, e.g., for process instance I: <*e1*, *e2*, *e3*, *e4*, *e5*, *e6*, *e7*, *e8*>. If events are non-overlapping, we merge subsequent start events and events, e.g., <*B started*, *B completed*, *A started*, *A completed*> is abbreviated by <*B*, *A*>.

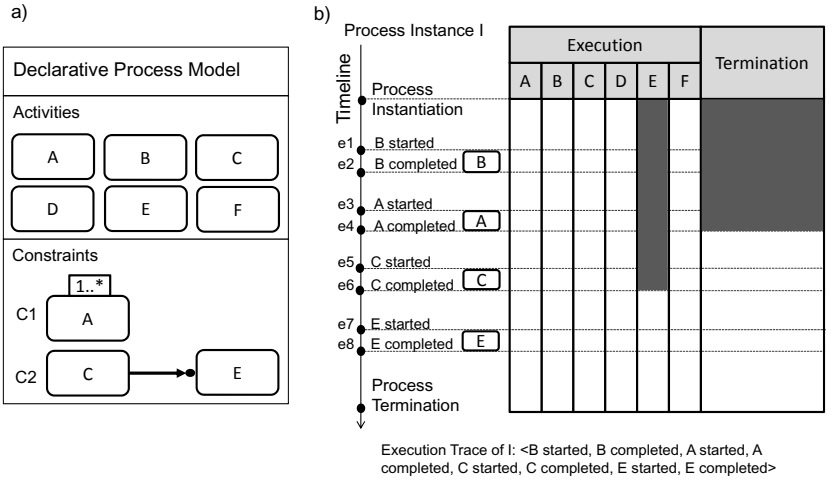


Fig. 1. Executing a Declarative Process

2.2 Shortcomings of Declarative Processes

While the declarative way of process modeling allows for a high degree of flexibility, this freedom comes at the cost of understandability problems and maintainability problems [12], [13], [14]. In particular, declarative process models are hard to read and understand, since the interactions between constraints quickly become too complex for humans to deal with [12]. Especially interactions that are not easily recognizable, so-called *hidden dependencies* [15], pose a significant challenge in reading and thus understanding declarative process models. Consider, for instance, the combination of cardinality constraints (i.e., an activity must be executed a specific number of times) and precedence constraints (i.e, an activity must be preceded by another activity) as illustrated in Fig. 2. Activity B has a cardinality of 1 (i.e., must be executed exactly once) and activity A is a prerequisite of B. Hence, in order to fulfill both constraints, A must be executed at least once. Since this interaction interaction is not *explicitly* visible, it is not sufficient that the modeler only relies on the information that is displayed explicitly, but has to carefully examine the process model for these hidden dependencies.

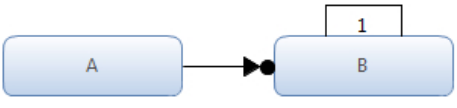


Fig. 2. Hidden Dependency

As discussed in [15], any change operation can be broken down into *sense-making tasks*, i.e., determining what to change and *action tasks*, i.e., perform the change. Declarative process models, as discussed, exhibit understandability issues that impede the sense-making task. This in turn hampers the action tasks and thus compromises the maintainability of declarative process models.

2.3 Test Driven Modeling

So far we discussed the benefits and drawbacks of declarative process models, now we briefly sketch how TDM is intended to support the maintenance of declarative models (for a detailed discussion we refer to [14]). A central aspect of TDM is the creation of so-called *testcases*. Testcases allow for the specification of behavior the process model must *exhibit* and to specify behavior the process model must *prohibit*. As the focus of TDM is put on control flow aspects, testcases provide mechanisms for the validation of control-flow related properties. In particular, a testcase consists of an *execution trace* (i.e., a sequence of events that reflect the current state of a process instance) as well as a set of *assertions* (i.e., conditions that must hold for a process instance being in a certain state). The execution trace thereby specifies behavior that must be supported by the process model, whereas assertions allow to test for unwanted behavior, i.e., behavior that must be prohibited by the process model. A typical example for an assertion would be to check, whether or not activity N is executable at time M.

Consider, for illustration, the testcase depicted in Fig. 3. It contains the execution trace $<A, B>$ (1) as well as an assertion that specifies that A cannot be executed between *e2* and *e3* (2) and assertions that specify that the process instance cannot be terminated before *e2* (3), however, it must be possible to terminate after *e2* (4).

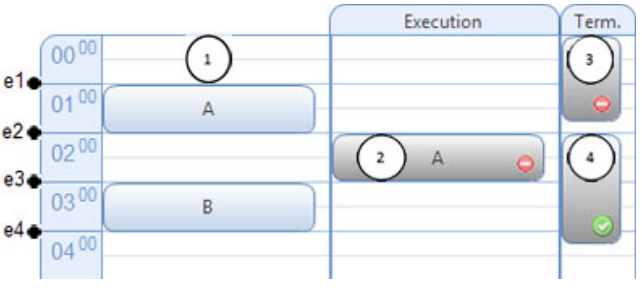


Fig. 3. A Simple Testcase

As illustrated in Fig. 3, testcases make information *explicit* that is only available in an implicit form in process models. For instance, in this example the process instance cannot be terminated until A has been executed, cf. termination assertion 3) and 4). Thus, testcases provide an *additional view* on the process model, which allows to resolve hidden dependencies by specifying testcases that make these dependencies explicit. Furthermore, it supports the interpretation of declarative process models and thus presumably lowers the cognitive load [14]. This additional view is not provided by a single testcase in isolation. Rather, a process model is combined with a set of testcases, where each testcase focuses on a specific part of the intended behavior only. With respect to maintenance, the close coupling of testcases and process model should help to ensure that changes conducted to the process model do not violate desired behavior (cf. regression testing in software engineering [16]). Similar to unit testing [17], testcases can be validated *automatically* by replaying the execution trace in a test environment and checking the assertions step-by-step. Testcases thus relieve modelers from checking *validity*, i.e., to test whether the process model appropriately reflects reality, manually, which presumably lowers the *cognitive load* of modelers and leads to quality improvements.

While it is known from software engineering that testcases indeed are able to improve perceived quality [18] and to improve quality [19], [20], the benefits of testcases for declarative process models so far are based on theoretical considerations only (cf. [14]). Whether or not these conjectures also holds for declarative process models we will investigate in the following.

2.4 Test Driven Modeling Suite

In order to enable the evaluation of TDM, Test Driven Modeling Suite (TDMS)[1] was implemented to provide the necessary operational support. In particular,

[1] Freely available from: `http://www.zugal.info/tdms`

TDMS provides an integrated development environment for the creation of testcases and declarative process models. In order to enable an in-depth analysis, TDMS was implemented on top of Cheetah Experimental Platform (CEP) [21]. In addition to the analysis of the *product*, i.e., the maintained process models, the generic replay feature of CEP allows to watch the *process of maintenance* step-by-step. Put differently, TDMS allows to inspect *each single step* the modeler undertook, thereby enabling researchers to investigate how the modeler approached the maintenance tasks and how the adoption of testcases influenced their behavior.

Fig. 4 shows a screenshot of a simple declarative model edited in TDMS. On the left hand side testcases are visualized (1); for this particular screenshot a testcase with execution trace $<A, B, B, B, A, C>$ and a termination assertion is shown. On the right hand side TDMS provides a graphical editor for designing the process model (2). Whenever a testcase or a process model are changed, TDMS immediately validates the testcases against the process model and indicates failed testcases in the testcase overview (3)—currently listing three testcases from which one failed. In addition, TDMS provides a detailed problem message about failed testcases in (4). In this example, the modeler defined that the trace $<A, B, B, B, A, C>$ must be supported by the process model. However, as A must be executed exactly once (cf. the cardinality constraint on A), the process model does not support this trace as indicated by the highlighted occurrence of activity A (1), the testcases marked in (3) and the detailed error message in (4). Since TDMS automatically validates all testcases whenever the process model is changed, the modeler is relieved from checking this control-flow behavior manually. Instead TDMS will automatically notify the modeler when she conducts changes that conflict with this requirement.

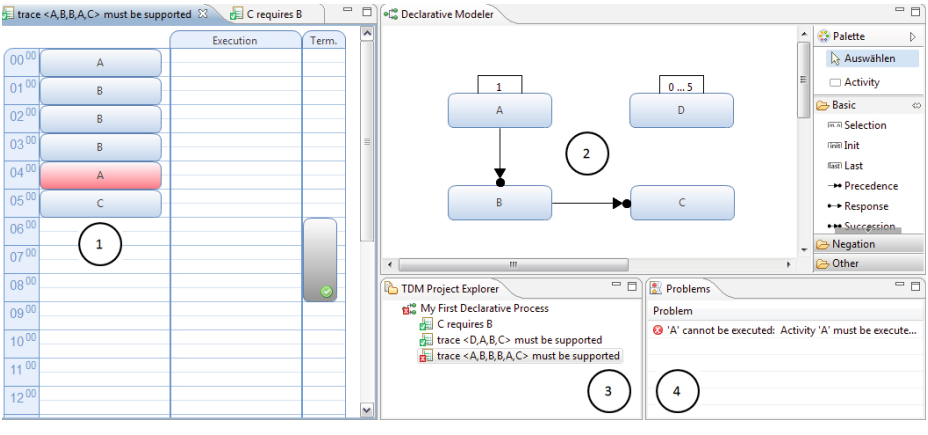


Fig. 4. Screenshot of TDMS

3 Experimental Definition and Planning

To test our theories, this section introduces the hypotheses, describes the subjects, objects, factors, factor levels and response variables of our experiment and presents the instrumentation and data collection procedure as well as the experimental design.

Hypotheses. The first hypothesis to be tested deals with the *cognitive load* of process modelers. Based on our theoretical work [14] we postulate that the adoption of testcases has a positive effect on the *cognitive load* of process modelers:

Hypothesis H_1***:*** *The adoption of testcases significantly lowers the cognitive load on the process modeler conducting the change.*

Secondly, we know from experiments conducted in the domain of software engineering that having testcases at hand improves perceived quality [18]. Similarly, we expect testcases to improve the perceived quality of the process model:

Hypothesis H_2***:*** *The adoption of testcases significantly improves the perceived quality of the adapted process model.*

Thirdly, testcases provide an automated way of validating the process model. Thus, we expect a positive influence on the quality of process models (the operationalization of quality will be explained subsequently):

Hypothesis H_3***:*** *The adoption of testcases significantly improves the quality of changes conducted during maintenance.*

Subjects. The targeted subjects should be at least moderately familiar with business process management and declarative process modeling notations. We are not targeting modelers who are not familiar with declarative process models at all, since we expect that their unfamiliarity blurs the effect of adopting testcases as they have to struggle too much with the notation itself.

Objects. The objects of our study are two change assignments, each one performed on a different declarative process model.[2] The process models and change assignments have been designed carefully to reach a medium level of complexity that goes well beyond the complexity of a "toy-example". To cancel out the influence of domain knowledge [22], we labeled the models' activities by letters (e.g., A to H). Furthermore, to counter-steer potential confusion by an abundance of different modeling elements, no more than eight *distinct* constraints have been used per model. In addition, we performed several pretests to ensure that the process models and change assignments are of appropriate complexity and are not misleading.

The change assignments consist of a list of requirements, so-called invariants, that hold for the initial model and *must not be violated* by the changes conducted. In addition, it must be determined, whether the change to be modeled is *consistent* with the invariants. If this is the case, the changes have to be performed while

[2] The material used for this study can be downloaded from: http://www.zugal.info/experiment/tdm

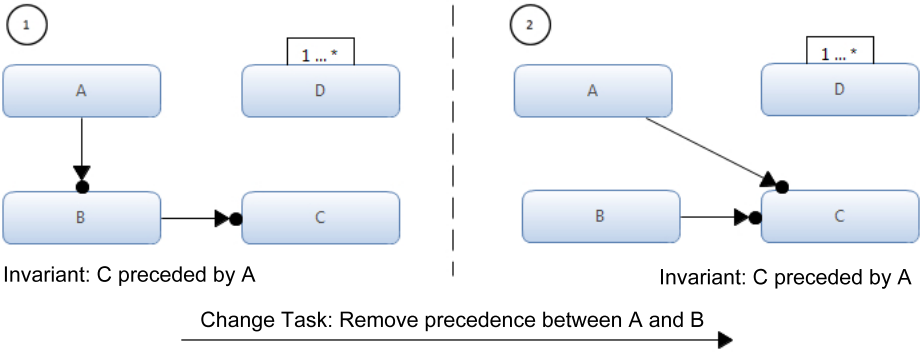


Fig. 5. Example of a Change Assignment

ensuring that all invariants are preserved. If a change assignment is identified to be inconsistent, a short explanation of the inconsistencies must be provided.

An example of a change assignment is illustrated in Fig. 5 (1). Assume an invariant that C cannot be executed until A has been executed. Further assume a change assignment to remove the precedence constraint between A and B. The invariant is valid for this model as C requires B to be executed before and B requires A to be executed before—thus C cannot be executed before A has been executed. The change is consistent, as it does not contradict the invariant. However, removing the precedence constraint between A and B is not enough. In addition, a new precedence constraint between A and C has to be introduced to satisfy the invariant, resulting in the process model shown in Fig. 5 (2).

Factor and Factor Levels. Our experiment's factor is the adoption of testcases, i.e., whether testcases are provided while conducting the changes to the process model or not. Thus, we define the factor to be *adoption of testcases* with factor levels *testcases* as well as *absence of testcases*.

Response Variables. In order to test the hypotheses formulated above, we define the following response variables: 1) *cognitive load* on the process modeler, 2) *perceived quality* as well as 3) *quality* of the process model. For measuring cognitive load and perceived quality, we ask subjects to self-rate their subjective perception. The measurement of quality is derived from the change assignments (cf. paragraph above discussing objects). In particular, we define quality to be the sum of *preserved (non-violated) invariants, the number of correctly identified inconsistencies as well as the number of properly performed changes*, i.e., we measure whether the *new* requirements have been modeled appropriately.

To illustrate this notion of quality, consider again the process model shown in Fig. 5 (1) and the change assignments from the paragraph discussing the objects. The modeler must 1) determine that the change is consistent, 2) remove the precedence constraint between A and B to fulfill the change assignment, as well as 3) introduce a new precedence constraint between A and C to satisfy the invariant—for each subtask one point can be reached, i.e., at most 3 points per change assignment.

Experimental Design. The experimental design is based on the guidelines for designing experiments in [23]. Following these guidelines, a *randomized balanced single factor* experiment is conducted with *repeated measurements*. The experiment is called *randomized*, since subjects are assigned to groups randomly. We denote the experiment as *balanced* as each factor level (i.e., the adoption of testcases and the absence of testcases) is applied to the same number of subjects. As only a single factor is manipulated (i.e., the adoption of testcases), the design is called *single factor*. Fig. 6 illustrates the described setup: the experiment is divided into two runs, whereas the objects (i.e., process models) are changed and the factor levels (i.e., adoption of testcases) are switched after the first run, thus achieving repeated measurements.

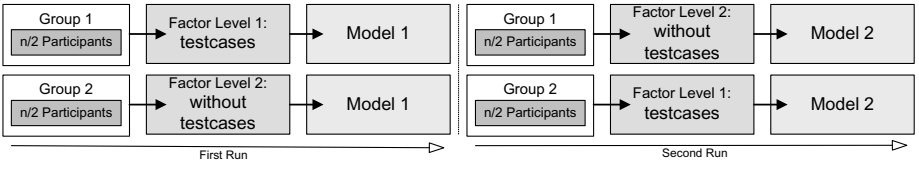


Fig. 6. Experimental Design

Instrumentation and Data Collection Procedure. As already pointed out, we rely on CEP for non-intrusive data collection. This, as detailed in [21], enables us to investigate the maintenance tasks in detail by replaying the logged commands step-by-step. Based on the log data, CEP's analysis capabilities provide additional methods for evaluation, e.g., for analyzing questionnaires or computing modeling metrics.

4 Performing the Experiment

This section deals with the experiment's execution. Section 4.1 covers operational aspects, i.e., how the experiment has been executed. Then, in Section 4.2 data is analyzed and subsequently discussed in Section 4.3.

4.1 Experimental Operation

Experimental Preparation. Preparation tasks included the implementation of TDMS and the elaboration of process models and change assignments. To ensure that the assignments are clearly formulated, several researchers with different backgrounds were included in the creation. Furthermore, we conducted pretests before the actual experiment to screen the assignments for potential problems.

Experimental Execution. The experiment was conducted in December 2010 at the University of Innsbruck in the course of a weekly lecture on business processes and workflows; all in all 12 students participated. To prepare the students, a lecture on declarative process models was held two weeks before the

experiment. In addition, students had to work on several modeling assignments using declarative processes before the experiment took place. One week before the experiment, the concept of testcases and their usage was demonstrated. Immediately before the experiment, a short lecture revisiting the most important concepts of TDM and the experiment setup was held. The rest of the experiment was guided by CEP's experimental workflow engine [21], leading students through an initial questionnaire, two modeling tasks (one with the support of testcases and one without the support of testcases), a concluding questionnaire and a feedback questionnaire. The experiment was concluded with a discussion to exchange students' experiences and to revisit the experiment's key aspects.

Data Validation. Due to the relatively small number of students participating, we were able to constantly monitor for potential problems or misunderstandings and to immediately resolve them. For this reason and owing to CEP's experimental workflow engine [21], all students have been guided successfully through the experiment—no single data set had to be discarded because of disobeying the experimental setup. In addition, we screened the subjects for familiarity with DecSerFlow [12] (the declarative process modeling language we use in our models), since our research setup requires subjects to be at least moderately familiar with DecSerFlow. The mean value for familiarity with DecSerFlow, on a Likert Scale from 1 to 7, is 3.17 (slightly below average). For confidence in *understanding* DecSerFlow models a mean value of 3.92 was reached (approximately average). Finally, for perceived competence in *creating* DecSerFlow models, a mean value of 3.83 (approximately average) could be computed. Since all values range about average, we conclude that the participating subjects fit the targeted profile.

4.2 Data Analysis

In the following we describe the analysis and interpretation of data.

Descriptive Analysis. To give an overview of the experiment's data, Table 1 shows minimum, maximum and mean values of cognitive load, perceived quality and quality. The values shown in Table 1 *suggest* that the adoption of testcases lowers cognitive load, increases perceived quality and increases quality, thus

Table 1. Descriptive Statistics

	N	Minimum	Maximum	Mean
Cognitive load with testcases	12	2	7	4.33
Cognitive load without testcases	12	4	7	5.75
Cognitive load overall	24	2	7	5.05
Perceived quality with testcases	12	4	7	6.00
Perceived quality without testcases	12	3	6	4.25
Perceived quality overall	24	3	7	5.12
Quality with testcases	12	20	23	22.33
Quality without testcases	12	19	23	21.92
Quality overall	24	19	23	22.13

supporting hypotheses H_1, H_2 and H_3. However, these observations are merely based on descriptive statistics. For a more rigid investigation, the hypotheses will be tested for statistical significance in the following.

Hypotheses Testing. Our sample is relatively small, thus we follow guidelines for analyzing small samples [24] to employ non-parametric tests. In particular, we use SPSS[3] to carry out Wilcoxon Signed-Rank Test [24].[4]

Hypothesis H_1: Applying Wilcoxon Signed-Rank Test for the response variable cognitive load yields a p-value of 0.010 (< 0.05), thus *supporting* H_1.
Hypothesis H_2: Applying Wilcoxon Signed-Rank Test for the response variable perceived quality yields a p-value of 0.005 (< 0.05), thus *supporting* H_2.
Hypothesis H_3: Applying Wilcoxon Signed-Rank Test for the response variable quality yields a p-value of 0.391 (> 0.05), thus *rejecting* H_3.

Summing up, hypotheses cognitive load (H_1) and perceived quality (H_2) are supported, while quality (H_3) had to be rejected. Reasons, implications and conclusions are discussed in the following.

4.3 Discussion

Based on the obtained analysis results we can conclude that the adoption of testcases has a positive influence on the cognitive load (H_1) and perceived quality (H_2). Especially interesting is the fact that, even though quality could *not* be improved significantly, apparently the modelers have been more confident that they conducted the changes properly. This effect is also known from software engineering, where testcases improve perceived quality [18], [25]. Indeed also the follow-up discussion with the students after the experiment revealed that students with software development background experienced this similarity, further substantiating our hypotheses on a qualitative basis.

Regarding quality (H_3) no statistically significant differences could be observed. This raises the question whether there is no impact of testcases on quality at all or if the missing impact can be explained otherwise. To this end, a detailed look at the distribution of quality offers a plausible explanation: the overall quality is *very high*, the quality measured on average is 22.13 out of a maximum of 23 (cf. Table 1). Thus, approximately 96% of the questions have been answered correctly / tasks have been carried out properly. Put differently, the overall quality leaves almost no room for improvements when adopting testcases—in fact, the sample's standard deviation is very low (1.39). Since data "points towards" the positive influence of testcases on quality (i.e., the mean value is higher, cf. Table 1) and due to the low variance it seems reasonable to assume that a positive correlation exists, however, the overall high quality blurs expected effects. To test this assumption, a replication with more complex and thus more challenging change tasks is planned. The increased complexity should result in a

[3] Version 17.0.
[4] Due to repeated measurements the variables are *not independent*, thus Wilcoxon Signed-Rank Test was chosen.

Table 2. Performed Change Operations

	N	Minimum	Maximum	Mean
Constraints created without testcases	12	5	13	9.83
Constraints created with testcases	12	10	34	16.25
Constraints deleted without testcases	12	5	12	8.42
Constraints deleted with testcases	12	9	32	15.17
Constraints adapted without testcases	12	0	1	0.33
Constraints adapted with testcases	12	0	7	0.92
Total without testcases	12	11	24	18.58
Total with testcases	12	19	73	32.33

lower overall quality, thereby leaving room for improvements and thus allow to distinguish the effect of adopting testcases.

As explained in Section 3, for the evaluation of quality we had to take a close look at the process models. Thereby, we could observe that the availability of testcases changed the behavior of the subjects. In particular, subjects who did not have testcases at hand seemed to be reluctant to change the process model, i.e., tried to perform the change tasks with a minimum number of change operations. To quantify and investigate this effect in detail, we counted 1) how many constraints were created, 2) how many constraints were deleted and 3) how many constraints were adapted. The results, listed in Table 2, reveal that modelers having testcases at hand approximately changed twice as many constraints. In fact, mean values computed are 18.58 versus 32.33. Wilcoxon Signed-Rank Test yields a p-value of 0.007, i.e., shows a *significant difference*. Again, we would like to provide a possible explanation that is inspired by insights from software engineering, where testcases improve the developer's confidence in the source code [17], [25]—and in turn increasing the developer's willingness to change the software. This experiment's data supports our assumption that a similar effect can be observed when declarative process models are combined with testcases—significantly more change operations were performed by modelers who had testcases at hand. And, even more interesting, the quality *did not decrease* even though approximately *twice as many* change operations have been performed. Thus, we conclude that testcases indeed provide an effective safety net, thereby increasing willingness to change while not impacting quality in a negative way.

Summing up, our experiment shows on the basis of empirical data that the adoption of testcases has a positive influence on the cognitive load, perceived quality as well as the willingness to change. Regarding the impact on quality, however, our data did not yield any statistically significant results. A closer look at the data suggests that these effects were blurred by the overall high quality. To clarify this issue we are currently preparing a replication including process models with higher complexity. Although the results sound very promising, we should not forget to mention that the small sample size (i.e., 12 subjects) constitutes a threat to *external validity*, i.e., it is questionable in how far the results can be generalized. However, it should be noted that if an effect can be shown

to be statistically significant in a small sample, it can be considered to be very strong [26].

5 Related Work

Most notably is the work of Ly et al. [27], which also focuses on the validation of the process model, however, in contrast to our work, adaptive process management systems are targeted instead of declarative ones. With respect to process validity, work in the area of process compliance checking should be mentioned, e.g., [28]. In contrast to our work, understandability of declarative languages is not of concern, the focus is put on imperative languages.

Another related stream of research is the verification of declarative process models. With proper formalization, declarative process models can be verified using established formal methods [12]. Depending on the concrete approach, a-priori (e.g., absence of deadlocks) [12] or a-posteriori (e.g., conformance of the execution trace) [29] checks can be performed. While these approaches definitively help to improve the *syntactical* correctness and provide *semantical* checks a-posteriori, they do not address understandability and maintainability issues.

Related are also so-called *scenario-based* approaches to process modeling, where scenarios specify a certain aspect of a business process similar to a testcase (e.g., [30], [31]). However, existing approaches focus on *imperative* process modeling notations, e.g., Petri Nets, whereas our approach is clearly focused on declarative process modeling notations.

With respect to empirical evaluation, experiments investigating the effect of Test Driven Development, i.e., interweaving software development and testing, like the TDM methodology interweaves modeling and testing [14], are of interest [19], [20], [18]. Since similarities between software processes and business processes are well known from literature [32], it is not surprising that similar result are reported (e.g., increased perceived quality, increased quality).

6 Summary and Outlook

Declarative approaches to business process modeling have attracted a recent interest as they promise to provide a high degree of flexibility [11]. However, the increase in flexibility comes at the cost of understandability problems and resulting maintainability problems of respective process models [12], [13], [14]. To compensate for these shortcomings, the TDM methodology adapts the concept of testcases from software engineering. While this approach seems beneficial in theory, no empirical evaluation has been provided yet. This paper picks up this need and contributes a controlled experiment investigating the impact of the adoption of testcases on the maintenance of declarative process models.

In particular, our experiment investigates the impact of testcases on the cognitive load, perceived quality and model quality. Our results show that the adoption of testcases allows for a significantly *lowered cognitive load* and *increased perceived quality*. For quality, however, no significant difference could be found,

presumably because of the only moderate complexity of process models. In addition, we could show that modelers who were supported by testcases performed approximately twice as many change operations (difference statistically significant). This effect is especially interesting as quality did not decrease, indicating that testcases are able to provide an effective safety net for maintenance tasks. However, while the results sound promising, their generalization is questionable due to the relatively small sample size of 12 participants.

To tackle this issue and to clarify whether a significant difference for quality can be observed for more complex change assignments, we are currently preparing a replication including more subjects and more complex assignments.

References

1. Lenz, R., Reichert, M.: IT support for healthcare processes - premises, challenges, perspectives. DKE 61(1), 39–58 (2007)
2. Poppendieck, M., Poppendieck, T.: Implementing Lean Software Development: From Concept to Cash. Addison-Wesley Professional, Reading (2006)
3. Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.: Process Aware Information Systems: Bridging People and Software Through Process Technology. Wiley Interscience, Hoboken (2005)
4. Weske, M.: Business Process Management: Concepts, Methods, Technology. Springer, Heidelberg (2007)
5. Reichert, M., Dadam, P.: ADEPTflex: Supporting Dynamic Changes of Workflow without Losing Control. JIIS 10(2), 93–129 (1998)
6. Weske, M.: Workflow Management Systems: Formal Foundation, Conceptual Design, Implementation Aspects. PhD thesis, University of Münster (2000)
7. van der Aalst, W.M.P., Weske, M.: Case handling: a new paradigm for business process support. DKE 53(2), 129–162 (2005)
8. Pesic, M., Schonenberg, H., Sidorova, N., van der Aalst, W.M.P.: Constraint-Based Workflow Models: Change Made Easy. In: Proc. CoopIS 2007, pp. 77–94 (2007)
9. 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)
10. Sadiq, S.W., Orlowska, M.E., Sadiq, W.: Specification and validation of process constraints for flexible workflows. ISJ 30(5), 349–378 (2005)
11. Weber, B., Reichert, M., Rinderle, S.: Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. DKE 66(3), 438–466 (2008)
12. Pesic, M.: Constraint-Based Workflow Management Systems: Shifting Control to Users. PhD thesis, TU Eindhoven (2008)
13. Weber, B., Reijers, H.A., Zugal, S., Wild, W.: The Declarative Approach to Business Process Execution: An Empirical Test. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 470–485. Springer, Heidelberg (2009)
14. Zugal, S., Pinggera, J., Weber, B.: Toward Enhanced Life-Cycle Support for Declarative Processes. JSME (to appear)
15. Green, T.R.G., Petre, M.: Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. JVLC 7(2), 131–174 (1996)
16. Agrawal, H., Horgan, J.R., Krauser, E.W., London, S.: Incremental Regression Testing. In: Proc. ICSM 1993, pp. 348–357 (1993)

17. Beck, K.: Test Driven Development: By Example. Addison-Wesley, Reading (2002)
18. Marchenko, A., Abrahamsson, P., Ihme, T.: Long-Term Effects of Test-Driven Development A Case Study. In: Abrahamsson, P., Marchesi, M., Maurer, F. (eds.) Agile Processes in Software Engineering and Extreme Programming. Lecture Notes in Business Information Processing, vol. 31, pp. 13–22. Springer, Heidelberg (2009)
19. Canfora, G., Cimitile, A., Garcia, F., Piattini, M., Visaggio, C.A.: Evaluating advantages of test driven development: a controlled experiment with professionals. In: Proc. ISESE 2006, pp. 364–371 (2006)
20. George, B., Williams, L.: A structured experiment of test-driven development. Information and Software Technology 46(5), 337–342 (2004)
21. Pinggera, J., Zugal, S., Weber, B.: Investigating the process of process modeling with cheetah experimental platform. In: Proc. ER-POIS 2010, pp. 13–18 (2010)
22. Khatri, V., Vessey, I., Ramesh, P.C.V., Park, S.-J.: Understanding Conceptual Schemas: Exploring the Role of Application and IS Domain Knowledge. Information Systems Research 17(1), 81–99 (2006)
23. Wohlin, C., Runeson, R., Halst, M., Ohlsson, M.C., Regnell, B., Wesslen, A.: Experimentation in Software Engineering: an Introduction. Kluwer, Dordrecht (2000)
24. Pett, M.A.: Nonparametric Statistics for Health Care Research: Statistics for Small Samples and Unusual Distributions. Sage Publications, Thousand Oaks (1997)
25. Beck, K.: Extreme Programming Explained: Embracing Change. Addison-Wesley, Reading (1999)
26. Royall, R.M.: The Effect of Sample Size on the Meaning of Significance Tests. The American Statistician 40(4), 313–315 (1986)
27. Ly, L.T., Rinderle, S., Dadam, P.: Integration and verification of semantic constraints in adaptive process management systems. DKE 64(1), 3–23 (2008)
28. Awad, A., Decker, G., Weske, M.: Efficient Compliance Checking Using BPMN-Q and Temporal Logic. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 326–341. Springer, Heidelberg (2008)
29. van der Aalst, W.M.P., de Beer, H.T., van Dongen, B.F.: Process mining and verification of properties: An approach based on temporal logic. In: Chung, S. (ed.) OTM 2005. LNCS, vol. 3760, pp. 130–147. Springer, Heidelberg (2005)
30. Fahland, D.: From Scenarios To Components. PhD thesis, Humboldt-Universität zu Berlin (2010)
31. Glinz, M., Seybold, C., Meier, S.: Simulation-Driven Creation, Validation and Evolution of Behavioral Requirements Models. In: Proc. MBEES 2007, pp. 103–112 (2007)
32. Osterweil, L.J.: Software processes are software too. In: Proc. ICSE 1987, pp. 2–13 (1987)

An Exploratory Approach to Process Lifecycle Transitions from a Paradigm-Based Perspective

Filip Caron and Jan Vanthienen

K.U.Leuven, Faculty of Business and Economics, Leuven Institute for Research on Information Systems (LIRIS), Naamsestraat 69, Leuven, Belgium
{Filip.Caron,Jan.Vanthienen}@econ.kuleuven.be

Abstract. While the use of a single business process paradigm (e.g. procedural or declarative) over the process lifecycle is often assumed in business process management, transitions between approaches at different phases in the lifecycle could also be examined. This paper explores several business process management strategies by analyzing the approaches at different phases in the process lifecycle as well as the various transitions between those phases.

Keywords: Business Process Management, Process Modeling, Process Enactment, Transitions.

1 Introduction

Multiple approaches to the control-flow perspective of business process management have been proposed in the literature, ranging from the procedural (e.g. [1]) to the declarative (e.g. [2]) business process paradigm with a series of hybrid paradigms in between (e.g. [3,4,5]). Each of these approaches proposes a different sets of tradeoffs between desirable process characteristics such as process compliance, flexibility, efficiency and effectiveness.

The use of one business process management approach over the full process lifecycle is often assumed. However, business processes may also require that different tradeoffs are made at different phases in the process lifecycle. For example, in practice it often occurs that at design-time the focus is placed on process compliance and on the traceability of the related directives, whereas process efficiency becomes the most important characteristic at run-time. The contribution of this visionary paper will be the exploration of various business process lifecycle strategies that allow the independent selection of both a design-time and a run-time process paradigm and that provide a transition path. These strategies may in practice result in a better fit between the business processes and the systems that support them.

This paper is structured as follows. Section 2 briefly characterizes the different strategic positions at each lifecycle phase. Section 3 introduces the transition strategies between the different design-time and run-time positions, which are discussed and evaluated in section 4. Finally, section 5 concludes the paper.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 178–185, 2011.

2 Positions at Design-Time and Run-Time

A traditional business process lifecycle consists of four phases with distinct roles (i.e. design, implementation/configuration, enactment and analysis phase). In the context of making business process management work, however, we distinguish two distinct strategy decisions: one at design-time and one at run-time. The process design-time consists of the design phase and all process analysis activities prior to the actual modeling. Process run-time, on the other hand coincides with the implementation and enactment phases.

In the business process management literature a wide spectrum of business process paradigms has been presented. These different paradigms can be roughly categorized into the following classes:

The ***procedural business process paradigm*** focuses on defining an exact activity sequence that will result in obtaining the related corporate goal (e.g. [1,6,7]).

The ***declarative business process paradigm*** focuses on capturing the regulatory and internal directives that impose restrictions on the business processes. Different declarative approaches have been proposed, such as the use of constraints, rules, event conditions or other (logical) expressions (e.g. [8,2]).

Hybrid business process paradigms focus on combining activity sequence specifications with declarative specifications, the principles of the other two business process paradigms (e.g. [5,9,10,11,12,13]).

The following subsections will further elaborate the different paradigm-based strategy options at the key phases in the process lifecycle.

2.1 Design-Time Positions

In the business process modeling stage, each business process paradigm has a different set of design principles and constructs, that will determine the characteristics of the final process model.

Procedural process models contain a precise definition of the control-flow, including the alternative execution paths, events and exceptions. Graph-based modeling languages (e.g. Petri Nets [1,7], BPMN [6] and EPC [14]) are commonly used for control-flow specifications [15].

Declarative process models specify process properties by means of event conditions (e.g. ECA-rules [16]), constraints (e.g. ConDec [17], DecSerFlow [18] and BPCN [4]) or other logical expressions.

Hybrid process models combine elements of both procedural and declarative process modeling techniques. Hybrid process modeling techniques generally make use of placeholder activities [5,9,10] or rule-based adaptation mechanism [11,12].

2.2 Run-Time Positions

Also at run-time a strong differentiation can be observed between the three business process paradigms. This section briefly describes the execution principles of each business process paradigm.

Procedural process enactment is characterized by a straightforward execution based on execution paths specified in a procedural process execution language such as BPEL [19] or YAWL [20].

Declarative process enactment is based on the dynamic run-time development of an execution scenario for each individual process instance. Formal declarative specifications [21,22] are used to impose restrictions on the dynamic scenario development. A multitude of techniques for the dynamic development of an execution scenario have been proposed, including dynamic planning algorithms (e.g. PLMflow) [23], ECA rules [16], event-driven BPM approaches [24] and LTL-automata [21].

Hybrid process enactment techniques are used to enact process models that contain placeholder activities. The base process is executed according to the principles of the chosen process paradigm and whenever a placeholder activity is encountered a switch in process paradigm takes place. Two approaches to switch enactment strategies have been proposed: the use of build-activities executed in the same workflow engine (e.g. Chameleon) [5] and the use of subprocesses encapsulated in a service [9].

3 Design-Time to Run-Time Transition Strategies

Traditional business process management solutions are oriented towards same-paradigm transitions, e.g. using procedural principles both at design-time and at run-time. These traditional solutions allow to fully exploit the advantages of a single process paradigm. However, in this paper we examine if cross-paradigm transitions might better support business processes that have different requirements at design-time and run-time, e.g. in terms of flexibility. In these cases we analyze design-time to run-time transitions between process paradigms, the cross-strategy paradigms (figure 1).

In addition to the same-paradigm transitions, three cross-paradigm transitions seem most interesting. Both the transition from hybrid modeling to procedural enactment and from declarative modeling to procedural enactment are characterized by a need for design-time flexibility and run-time efficiency. The transition from procedural modeling to declarative enactment could be motivated by the need for a distributed implementation at run-time combined with a clean process overview at design-time. Other cross-paradigm transitions could rely on splitting up the business processes and performing different transition types on different process parts (and thus are not shown in figure 1), e.g. a hybrid to declarative transition combines procedural to declarative and declarative to declarative transition principles for different process parts.

3.1 Same-Paradigm Transitions

Procedural-procedural transition. Different *transformation strategies* between procedural process modeling languages and procedural process execution languages have been proposed in the literature [25,26]. Due to a conceptual mismatch between the standard procedural process modeling languages and execution

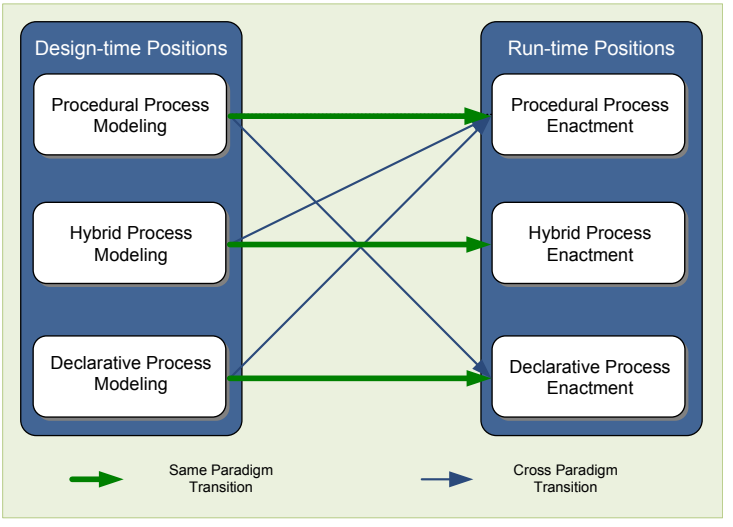


Fig. 1. Design-time to run-time transitions

languages, translation techniques are only offered for process models captured in a core subset of the procedural process modeling languages [27,25]. Additionally, the *process variant management* approaches based on querying a repository of procedural process variants (e.g. [4]), fall into the scope of this transition type.

Declarative-declarative transition. The common declarative process modeling languages have their roots in *formal logic*, e.g. ConDec provides a graphical notation for specific LTL expressions [21]. Due to this formal foundation, the translation of a high-level declarative process model into enactable rules is rather straightforward. These enactable rules are then used to govern declarative process instances by non-deterministic workflow engines (i.e. rule-based event-driven process engine).

Hybrid-hybrid transition. The hybrid-hybrid transition can only be applied on hybrid process models that contain placeholder activities. For each process part (the base process or a placeholder activity) the corresponding same-paradigm transition is selected.

3.2 Cross-Paradigm Transitions

Procedural-declarative transition. The procedural process model is translated into a set of event-based rules (e.g. preconditions), which can be used for a declarative process enactment [28]. While the focus is mostly placed on the translation of the control-flow, Dumas et al. describe an approach to deal with the data-flow in UML activity diagrams [29].

Declarative-procedural transition. Before run-time a systematic procedure is used for the construction of an optimal control-flow with reference to a particular characteristic. The systematic procedures for constructing an optimal

control-flow from a declarative process model are closely related to artificial intelligence planning techniques [30,31,32,33].

Hybrid-procedural transition. Within the context of this transition the focus is primarily placed on hybrid models of the second type, the process models that combine a full procedural specification with a set of business rules. Before run-time the procedural reference model is *customized to the specific needs* of a particular case by applying the set of customizing business rules [11,3]. The hybrid-procedural transition also occurs in the context of hybrid process models with placeholder activities, when the declarative-procedural transition is used for the declaratively specified process parts.

4 Discussion of the Different Transition Types

This section further elaborates on the different design-time to run-time transition types. Additionally, the different transition types are evaluated based on their impact on the process characteristics.

4.1 When Do Same-Paradigm Transitions Work?

Procedural-procedural transitions can be recommended for business processes in a *stable environment with predictable execution paths*. Under these conditions optimal process efficiency could be achieved. Repository-based variant management allows for anticipated process flexibility, all anticipated execution paths are specified. Stable environments are advisable since managing evolutions in the business concerns can be challenging in the context of the procedural paradigm [34], especially for repositories with large collections of process variants. Typical use cases are business processes for processing standard and static items, such as online orders or standardized financial transactions.

Declarative-declarative transitions are suitable for business processes in a *highly evolving environment* and/or business processes with *non-predictable execution paths*. Maximum process flexibility can be obtained [35]. Process compliance is guaranteed when all regulation and business policies are correctly mapped onto mandatory business constraints. As declarative process management systems might provide limited support at run-time [36], this transition type will be most suited for experts dealing with unique cases [37]. A typical use case for this transition would be non-standardized healthcare processes: while general medical principles are the same for all patient, each case will be different due to complications, patient conditions, etc.

Hybrid-hybrid transitions will be used for business processes that contain both process parts with *stable and highly evolving environments* or that consist of both process parts with *predictable and non-predictable execution paths*. The transition principles for individual process parts are similar to the corresponding same-strategy transition. However, it should be noted that process variant management approaches based on hybrid specifications will be easier to maintain than repository based transitions, because the procedural process parts are

not duplicated. A typical case for the hybrid-hybrid transition is a process that supports an advisory project of a consulting company. These processes combine very structured process parts, such as administrative activities at the beginning and the end, with an unstructured set of problem-solving activities in the middle.

4.2 When Could Cross-Paradigm Transitions Work?

Procedural-declarative transitions *slightly increase the run-time process flexibility* compared to the procedural-procedural transition. This increase results from the possibility of replacing or adding service task at run-time and the ability to define extra event-based rules at run-time to deal with temporary circumstances [29]. However, control-flow dependencies in the procedural process model that are not dictated by internal or external directives (i.e. the issue of overspecification), will still be mapped on event-based business rules and consequently impose restrictions on the run-time flexibility.

Declarative-procedural transitions combine extensive *process flexibility at design-time* with *process execution efficiency at run-time*. The run-time flexibility remains limited to the flexibility offered by procedural enactment. However, the declarative process model in combination with a time-efficient planning algorithm, allows for a *rapid adoption of new compliance requirements*. This is useful for processes that require far-reaching redesigns at regular intervals, but are at the same time relatively stable in the periods between those redesign phases.

The use of an artificial planning algorithm might positively affect both the *process efficiency and effectiveness*, since an optimization criterion needs to be specified. In addition, compared to declarative process enactment the end-user will be sufficiently *guided* and supported [38].

Hybrid-procedural transitions provide a neat approach to *process variant management* is provided. Compared to the process variant management approach introduced in the procedural-procedural transition, *maintenance of requirements is not needlessly complicated* since there is no duplication of the base process. However, the customization of the process model must be performed correctly and completely in order not to affect the process effectiveness.

5 Conclusion

Designing information systems that provide optimal support for specific business processes can be a challenging task. This paper promotes a clear distinction between the business process strategies and their differences at distinct stages in the process life cycle. The optimal selection of design-time and run-time positions (and consequently the transition type) will be process-specific and based on the business environment. Future research in business process strategies will be mainly focused on the further identification and exploration of typical use cases for the cross-paradigm transitions and evaluation of the proposed techniques.

References

1. Zisman, M.: Representation, specification and automation of office procedures. PhD thesis, Wharton School (1977)
2. van der Aalst, W., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science-Research and Development 23(2), 99–113 (2009)
3. Hallerbach, A., Bauer, T., Reichert, M.: Configuration and management of process variants. In: Handbook on Business Process Management 1, pp. 237–255. Springer, Heidelberg (2010)
4. Lu, R., Sadiq, S., Governatori, G.: On managing business processes variants. Data & Knowledge Engineering 68(7), 642–664 (2009)
5. Sadiq, S., Orlowska, M., Sadiq, W.: Specification and validation of process constraints for flexible workflows. Information Systems 30(5), 349–378 (2005)
6. Group, O.M.: Business process modeling notation (bpmn) 1.1 (2006)
7. Ellis, C., Nutt, G.: Modeling and enactment of workflow systems. In: Application and Theory of Petri Nets 1993, pp. 1–16 (1993)
8. Goedertier, S., Vanthienen, J.: In: An Overview of Declarative Process Modeling Principles and Languages. Communications of systemics and informatics world network, vol. 6, pp. 51–58 (2009)
9. van der Aalst, W.M.P., Adams, M., ter Hofstede, A.H.M., Pesic, M., Schonenberg, H.: Flexibility as a service. In: Chen, L., Liu, C., Liu, Q., Deng, K. (eds.) DASFAA 2009. LNCS, vol. 5667, pp. 319–333. Springer, Heidelberg (2009)
10. Schonenberg, H., Mans, R., Russell, N., Mulyar, N., Aalst, W.: Process flexibility: A survey of contemporary approaches. In: Advances in Enterprise Engineering I, pp. 16–30 (2008)
11. Kumar, A., Yao, W.: Process Materialization Using Templates and Rules to Design Flexible Process Models. In: Proceedings of the 2009 International Symposium on Rule Interchange and Applications, pp. 122–136. Springer, Heidelberg (2009)
12. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the Provop approach. Journal of Software Maintenance and Evolution: Research and Practice
13. Sinur, J.: The art and science of rules vs. process flows. Gartner Research, ID Number G00166408 (March 2009)
14. Keller, G., Nuttgens, M., Scheer, A.: Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK). Inst. fur Wirtschaftsinformatik (1992)
15. Recker, J., Rosemann, M., Indulska, M., Green, P.: Business process modeling: a comparative analysis. Association for Information Systems (2010)
16. Kappel, G., Rausch-Schott, S., Retschitzegger, W.: Coordination in workflow management systems a rule-based approach. In: Coordination Technology for Collaborative Applications, pp. 99–119 (1998)
17. Pesic, M., van der Aalst, W.: A declarative approach for flexible business processes management, pp. 169–180 (2006)
18. van der Aalst, W., Pesic, M.: DecSerFlow: Towards a truly declarative service flow language. In: Web Services and Formal Methods, pp. 1–23 (2006)
19. OASIS: Web services business process execution language 2.0 (2007)
20. van der Aalst, W., Ter Hofstede, A.: YAWL: yet another workflow language. Information Systems 30(4), 245–275 (2005)

21. Pesic, M., Aalst, W., Eijnatten, F.: Constraint-based workflow management systems: shifting control to users. PhD thesis, Technische Universiteit Eindhoven (2008)
22. Yu, J., Manh, T.P., Han, J., Jin, Y., Han, Y., Wang, J.: Pattern based property specification and verification for service composition. In: Aberer, K., Peng, Z., Rundensteiner, E.A., Zhang, Y., Li, X. (eds.) WISE 2006. LNCS, vol. 4255, pp. 156–168. Springer, Heidelberg (2006)
23. Zeng, L., Flaxer, D., Chang, H., Jeng, J.: PLM flow—Dynamic Business Process Composition and Execution by Rule Inference. In: Technologies for E-Services, pp. 51–95 (2002)
24. Paschke, A., Boley, H.: Rules capturing events and reactivity. In: Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches, IGI Publishing (2009)
25. Decker, G., Dijkman, R., Dumas, M., García-Bañuelos, L.: Transforming BPMN diagrams into YAWL nets. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 386–389. Springer, Heidelberg (2008)
26. Ouyang, C., Dumas, M., Aalst, W., Hofstede, A., Mendling, J.: From business process models to process-oriented software systems. ACM transactions on software engineering and methodology (TOSEM) 19(1), 1–37 (2009)
27. Recker, J., Mendling, J.: On the translation between BPMN and BPEL: Conceptual mismatch between process modeling languages. In: Proceedings of 18th International Conference on Advanced Information Systems Engineering (2006)
28. Casati, F., Ceri, S., Pernici, B., Pozzi, G.: Deriving active rules for workflow enactment. In: Database and Expert Systems Applications, pp. 94–115. Springer, Heidelberg (1998)
29. Dumas, M., Fjellheim, T., Milliner, S., Vayssière, J.: Event-based coordination of process-oriented composite applications. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 236–251. Springer, Heidelberg (2005)
30. El Maghraoui, K., Meghranjani, A., Eilam, T., Kalantar, M., Konstantinou, A.V.: Model driven provisioning: Bridging the gap between declarative object models and procedural provisioning tools. In: van Steen, M., Henning, M. (eds.) Middleware 2006. LNCS, vol. 4290, pp. 404–423. Springer, Heidelberg (2006)
31. Hendler, J., Tate, A., Drummond, M.: AI planning: Systems and techniques. AI Magazine 11(2), 61 (1990)
32. Friedler, F., Tarjan, K., Huang, Y., Fan, L.: Combinatorial algorithms for process synthesis. Computers & Chemical Engineering 16, S313–S320 (1992)
33. Liu, J., Fan, L., Seib, P., Friedler, F., Bertok, B.: Downstream process synthesis for biochemical production of butanol, ethanol, and acetone from grains: Generation of optimal and near-optimal flowsheets with conventional operating units. Biotechnology Progress 20(5), 1518–1527 (2004)
34. Fahland, D., Mendling, J., Reijers, H., Weber, B., Weidlich, M., Zugal, S.: Declarative vs. Imperative Process Modeling Languages: The Issue of Maintainability. In: 1st International Workshop on Empirical Research in Business Process Management, Ulm, Germany, pp. 65–76 (2009)
35. Regev, G., Soffer, P., Schmidt, R.: Taxonomy of flexibility in business processes. In: Regev, G., Soffer, P., Schmidt, R. (eds.) BPMDS (2006)
36. Weber, B., Sadiq, S., Reichert, M.: Beyond rigidity–dynamic process lifecycle support. Computer Science-Research and Development 23(2), 47–65 (2009)
37. Schmidt, R.: Flexibility in Service Processes. In: BPMDS 2006 (2006)
38. Bider, I.: Masking flexibility behind rigidity: Notes on how much flexibility people are willing to cope with. In: Proceedings of the CAiSE, vol. 5, pp. 7–18 (2005)

TTMS: A Task Tree Based Workflow Management System

Jens Brüning and Peter Forbrig

University of Rostock, Department of Computer Science,
Albert Einstein Str. 21,
D - 18059 Rostock, Germany
{jens.bruening,peter.forbrig}@uni-rostock.de

Abstract. In this paper, an approach to use task trees in a workflow management system (WfMS) is presented. As hierarchical models task trees capture several hierarchy levels of a workflow in one model. A workflow editor visualizes the models also as flowcharts similar to UML activity diagrams. The WfMS use these models as input and instantiates and executes them. The system is web-based and can be easily accessed by users with any browser clients. This paper motivates the approach to use task trees that produce hierarchical and structured workflow specifications. The proposed language might help end-users to better understand workflow models with its problem oriented hierarchical modeling character. Temporal operators from the task models are compared with certain operators from established workflow languages. In addition, in TTMS an instantiation time concept is implemented, where decision operators are evaluated at the very moment the process is instantiated. Consequently the task tree modeling language is enhanced for modeling decisions in the context of workflow management.

Keywords: User-oriented business process modeling, Business process management, Workflow systems, Task modeling.

1 Introduction

Nowadays, workflows are frequently modeled in a non-hierarchical, flat way in EPCs, UML activity diagrams or BPMN. Although subprocesses can be defined, the hierarchical modeling is not an integral part of these languages and can only be integrated in an unnatural, difficult to handle way. Having several hierarchies would mean managing several models and inserting a new hierarchy would mean creating new models. Whereas hierarchy is an integral part in function trees that are used for software engineering [1] to capture functional requirements and also in the ARIS method and toolkit [20] for business process modeling. Hierarchical supply and value chains, e.g. in the SCOR model [18] are a common feature in the domain of business modeling. Task models [6] are used for capturing the hierarchical character of tasks and get expressed in a tree-like notation. Although these three modeling languages have a similar hierarchical modeling concept they are used for different purposes. Figure 1 illustrates three examples, one for each language.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 186–200, 2011.

1.1 Hierarchical Modeling

Figure 1a) illustrates a part of the hierarchical SCOR model [18] in which the business process is the central part in the model. Figure 1b) visualizes a function tree in which the root node represents the main function of a software system or a service. The starting point for modeling is the system (the root node), which is refined into sub-functions and subsystems. This modeling perspective is system centric. In contrast, task models are used in the field of Human Computer Interaction (HCI) for modeling tasks and workflows in a user centered way [6]. The starting point is the root node of the task tree which describes a human goal, problem or task. The following sublevel in the hierarchy of the task tree describes how the goal is achieved by declaring the required tasks to be accomplished. Thus, the model is developed top down. Task trees also specify control flow within in contrast to function trees. Several languages have been developed for task models. The first one was HTA (Hierarchical Task Analysis) [6] followed by GOMS [6, pp.83-117] and CTT [17] and several others. CTT seems to be the most prominent one for task modeling.

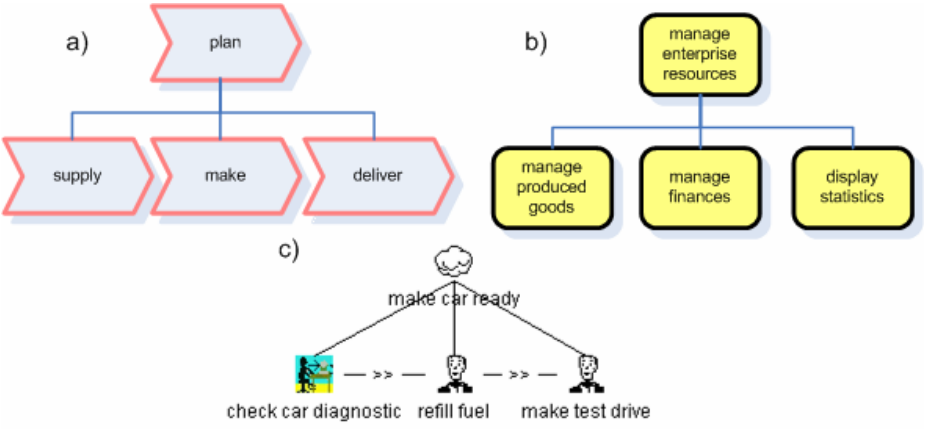


Fig. 1. a) Hierarchical SCOR model, b) function tree for a ERP software, c) task tree in CTT notation

The strict hierarchical structure of the workflow models can express the goals very well. By navigating one level upwards in the hierarchy the context goal of the current task can be seen. For example in Figure 1c) the context goal of the task *check car diagnostic* is the root task *make car ready*. Hierarchies for modeling goals are also used for example in the ARIS method [20] within hierarchical goal diagrams in combination with function trees for capturing the functional modeling part of the ARIS house.

The hierarchical structure answers the question of how a task or a problem can be handled by visiting child nodes. Using hierarchies for that purpose is very intuitive [15]. It is also discussed in the context of goal and problem modeling with a business extension of UML [7, pp. 99-105]. It is illustrated in Figure 1c) where the subordinate hierarchy level explains how the task or problem *make car ready* has to be handled.

The downward navigation in the tree stops at the leaves of the tree where the user action level is reached.

1.2 Task Models for Workflow Modeling

Task models can also be used for modeling business processes or workflows which will be the main contribution of this paper. In task trees workflows are modeled in a human centered way by focusing on the human goal or task at hand as root node. Within business processes not the human task or goal is relevant but one of the company or department. This is discussed more deeply in subsection 2.1. Nevertheless, while modeling the processes at design time or within the workflow execution in the WfMS at runtime the strict hierarchical character and presentation of the workflow models might help the end-users to understand the processes better than the ones modeled in a flat way by presenting the context goals in the model.

Task trees are structured workflow models [11] that provide further benefits. In [4] the correlation between CTT and UML activity diagrams is demonstrated. The structuredness within single-entry-single-exit regions [21] of every activity can be seen. Structured workflow models are less error prone, better readable and understandable [14]. Unreadable and unstructured workflow models similar to "Spaghetti code" [8] are not allowed. Although expressiveness is lost compared to the flowchart oriented languages like BPMN, there are many benefits to use task trees as workflow models. Consequently it seems promising to use them by a WfMS to instantiate and manage the work in an organization or company. A system implementing this approach is presented in this paper.

The rest of the paper is structured as follows. In section 2 the foundation for modeling workflows or rather business processes with task trees is given. Task modeling using CTT and TTMS is presented. The task modeling approach is enriched by an explicit decision modeling concept. This is mandatory for a WfMS to support the end-user in making decisions during the workflow execution. Additionally, it includes an instantiation time concept for these operators. Section 3 introduces the Task Tree based Workflow Management System (TTMS WfMS). We show how TTMS workflow models are interpreted and managed by the system. Related work will be presented in section 4 and the paper concludes with the summary in section 5.

2 Modeling Workflows with Task Trees

This section introduces the modeling of workflows with task trees. The approach behind TTMS workflow models is similar to CTT models. Hierarchy is an integral part of the workflow specifications and binary temporal relations are used to specify the control flow. Temporal operators are based on the operators of the process algebra LOTOS that is the logical fundament of CTT models [15]. TTMS accepts only a subset of CTT operators. But additionally, TTMS provides explicit choice operators inspired by the explicit decision modeling of EPCs. Finally, the development process of task models in CTTE and the editor for the TTMS system is introduced. A tool chain for producing workflow models and using them in the TTMS system is proposed.

2.1 Control Flow Specification Using Temporal Operators

CTT is probably the most popular language for task modeling. Figure 2a) and b) show examples of CTTs. Tasks are decomposed into subtasks within the tree. They are interrelated by binary temporal operators. Beside the binary temporal operators there are unary ones that are assigned to just one task. The CTT operators that are also part of TTMS are listed in Table 1. A complete listing and explanation of the CTT operators can be found in [17]. CTT priority order for the temporal operators can be seen in [15]. This order is important for interpreting different operators in the same level as pictured in Figure 2b). The priority of the interleaving (|||) operator is higher than the enabling operator (>>) so that the model is interpreted like the BPMN diagram of Figure 2c). The only unary operator listed in Table 1 is the iteration. In TTMS a little extension of the CTT iteration exists. With the *n* in the TTMS a count number can be specified at design time that declares the number of iterations of that specific task. Unbound iterations are specified with a ' * '.

Table 1. Temporal operators of CTT and TTMS

Operator name	CTT Notation	TTMS Notation
enabling	T1 >> T2	T1 ; T2
choice	T1 [] T2	T1 | T2
interleaving	T1 ||| T2	T1 + T2
iteration	T1*	T1 {n}

Before we describe how task trees can be used not only in a human but rather in a business oriented way we shortly introduce the cooperative CTT (CCTT) [17] modeling approach.

The starting point for task modeling as a human centered approach is the root node that describes a human task, goal or problem. In CCTT a role name is assigned to root nodes. A set of tasks or rather task trees describe the responsibilities of the role. The role can also describe a position in a company. The cooperation to tasks related to other roles is specified in a separate tree that represents a cooperative task or goal its root node. The cooperative task tree is refined until tasks are reached that are related to a specific role [15].

A way to use task trees in a more business oriented way is pictured in Figure 2a). The tree has the company as the root node and then first the organizational level is modeled in the tree similar to an organizational chart. Afterwards the related business processes are modeled in the business process level. The tasks modeled directly under the departments describe their responsibilities. Cooperation within the processes between subunits or divisions of the company cannot be specified in one tree because of its inherent structure. Considering the example of Figure 2a), a process of the *sell service* has no effects on any process of the *technical service*. Cooperative parts have to be specified in separate trees describing the cooperative tasks in CCTT, like described above. In contrast, BPMN uses message flows to model the cooperation between business partners or departments in a company.

The business process level ends after a certain point of abstraction. That is reached when activities cannot be sensibly further divided for business or economic reasons

[10]. The task modeling level goes beyond the business process level. The further modeling downwards is naturally supported by the integral hierarchical character of the task models. In the most cases it is a smooth transition from the business process level to the task level. An indication for the crossing between these two levels exists when lower tasks are related to users and upper tasks namely context goals are related to the company or department.

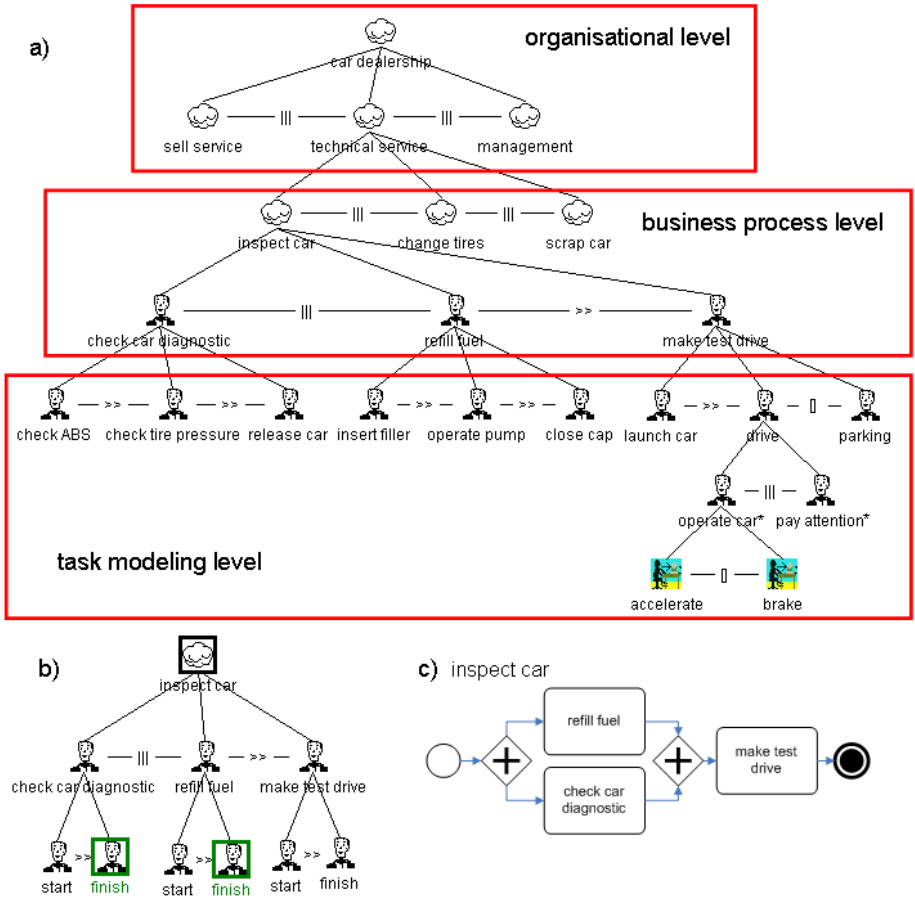


Fig. 2. a) an integrated task tree with different abstraction levels, b) a workflow model in execution in CTTE, c) The process *inspect car* in a structured BPMN chart

In the model of Figure 2a) the car dealership company has three subunits: *sell service*, *technical service* and *management*. Business processes are assigned to the organizational units. In the case of Figure 2a) the processes *inspect car*, *change tires* and *scrap car* are assigned to the *technical service*. The process *inspect car* is refined in the tree within the business process level. This process contains three activities that are in the temporal relationship as modeled in Figure 2b) and c). Below the process

level the task modeling level specifies the actions of the activities. The sequential operator is frequently used to specify the action sequences.

The *drive* task is further refined in the task modeling level of the task tree. The iterative task *operate car* is specified by repeatedly choosing the brake or accelerate actions after the deferred choice principle [23]. Additionally, the *pay attention* task is specified interleaved to *operate car*. These iteration tasks stops after the task *parking* is selected for execution. The listed actions would not be part of a business process model that is represented by the model managed by a WfMS. Thus, task modeling considers fine granular actions in contrast to business process modeling.

Figure 2b) shows a task tree model as a business process or rather workflow model. That will be the relevant kind of models to be considered in this paper for TTMS. In the CTT model of Figure 2b) explicit *start* and *finish* events are related to every activity connected with an *enabling* operator. The execution of the activity is left to the user to happen in between these events. Because of the hierarchical character task models can be easily refined into an arbitrary fine granular atomic action level. Explicit *start* and *finish* events are normally not needed for that level of abstraction. This has to be considered while modeling business processes with task models.

YAWL (Yet another workflow language) is a workflow language which is supported by a WfMS [24]. *start* and *finish* events are also used in YAWL for every activity or rather workitem [24]. YAWL is based on low level Petri net theory and uses explicit *start* and *complete* transitions with an *exec* place in between for that [24, Figure 10]. The actions of the user are left unspecified during the execution of the task in the WfMS. To resemble that concept with CTT models and thus model and simulate the WfMS model execution in the *CTT Environment* (CTTE) the model of Figure 2b) is used. CTTE is a powerful and mature tool to model and analyze task trees. The dynamic properties of the models can be checked by executing the task models within the CTTE tool. A task model in execution is pictured in Figure 2b). Another feature of CTTE is the enabled task list that is provided for the task model animation in which all enabled tasks are shown [17]. This concept is similar to *worklists* in WfMS, which include all enabled workitems [9, 24].

The *start* and *finish* events must not be modeled by the TTMS editor similar to the YAWL workflow editor. They are also omitted in the business process models like the one of Figure 2c) in BPMN. They are rather considered within the workflow execution in the WfMS like it is discussed in subsection 3.2.

Figure 2c) shows a structured workflow model in BPMN notation that describes the same process as in Figure 2b).

2.2 Different Choice Operators

The workflow language introduced in this paper uses a subset of the CTT language but adds some important operators for decision modeling. As discussed in the subsection before, CTT is normally used for modeling top-down to the user action level. The CTT choice modeling is more related to the deferred choice pattern which is implicit and based on events [23]. It is less based on a user decision modeled in an activity which is the semantics of decision modeling within EPCs [10, 3]. The CTT choice (T1 [] T2) is interpreted at action level. Thus, if T1 is a composed task and one of its enabled atomic action is performed T2 and its whole subtasks are skipped implicitly.

The CTT decision operator is insufficient to be used in a WfMS, where explicit decisions are needed. On the one hand the end-user has to know when a decision has to be made and on the other hand she has to be supported to choose the appropriate path in the workflow.

In [12] decision nodes have been proposed to be used in task trees for specifying decisions for model configuration. They have to be made before model execution in a preprocessing step to adept the task model to the *context of use*. If only decision nodes are used in the task model a decision tree model [1] is specified. TTMS uses these decision nodes for explicit decision modeling. The preprocessing step for interpreting the decision node happens at runtime. When the interpreter finds a decision node the user has to select the respective subtrees before these are further interpreted. Thus, not only an XOR decision but also a multi-choice (OR) decision [23] is integrated. In Figure 3a) a decision node is shown in a CTT-like notation which is similar to the notation used in [12]. But in contrast, the type of the choice (XOR or OR) has to be denoted there.

In the model of Figure 3a) and 3c) the activity *check car diagnostic* is a decision node. It is similarly interpreted as a decision activity in EPCs that can be seen in the activity *Check car diagnostic* in Figure 3b). A decision node in TTMS as well as the decision function in EPCs has a name that describes the activity in which the decision has to be made. In Figure 3 an OR decision is modeled so that the user must select one or more paths at runtime. The alternatives presented to the user during execution of the decision node are modeled in the guards connected to the respective subtrees in Figure 3a) and c).

In workflow modeling languages like BPMN, YAWL or UML activity diagrams the explicit choice operator is data-driven and automatically interpreted by the system. In the TTMS approach the explicit choice operator is integrated in a process model and is user-driven without the need of access to an external data model. This interpretation of decision modeling is more related to EPCs as it is pictured in Figure 3b). The decision activity represents the decision making task and stands in front of the choice connector. Related events modeled just behind the connector represent the criteria for choosing the desired paths in the process model [3].

Furthermore, Figure 3b) shows the semantics of the OR join. As introduced before, we have a structured process model by the tree structure of the model in Figure 3a). Thus, a related OR split exists to the OR join that is shown in Figure 3b). The semantics for the OR join is defined with the *Structured Synchronization Merge* (WCP7) of the workflow patterns [23].

Nevertheless, besides the explicit choice operators the standard CTT choice that behaves like a deferred choice is still supported within this approach although the explicit choice operator is definitely more relevant for workflows guided by a WfMS.

In combination with the implementation of the explicit choice operators an instantiation time concept is integrated in TTMS. This concept is described in the context of EPCs in [2, 3] and is also mentioned in [22]. With the process instantiation all the instantiation time operators are evaluated by the WfMS. The interpretation by the WfMS and the user interface generated from these operators are described in subsection 3.1. Similarly to [13] questions to configure the process model are asked. The questions are the labeling of the decision node. The criteria for selecting the correct subtrees are modeled in guards connected to the respective subtrees. In contrast to

C-EPCs [19] and C-YAWL [13] the decision is made by the user who is asked from the WfMS just before beginning of the process execution. The user must make the decision or can defer it into runtime to that particular point in the process model where the operator stands.

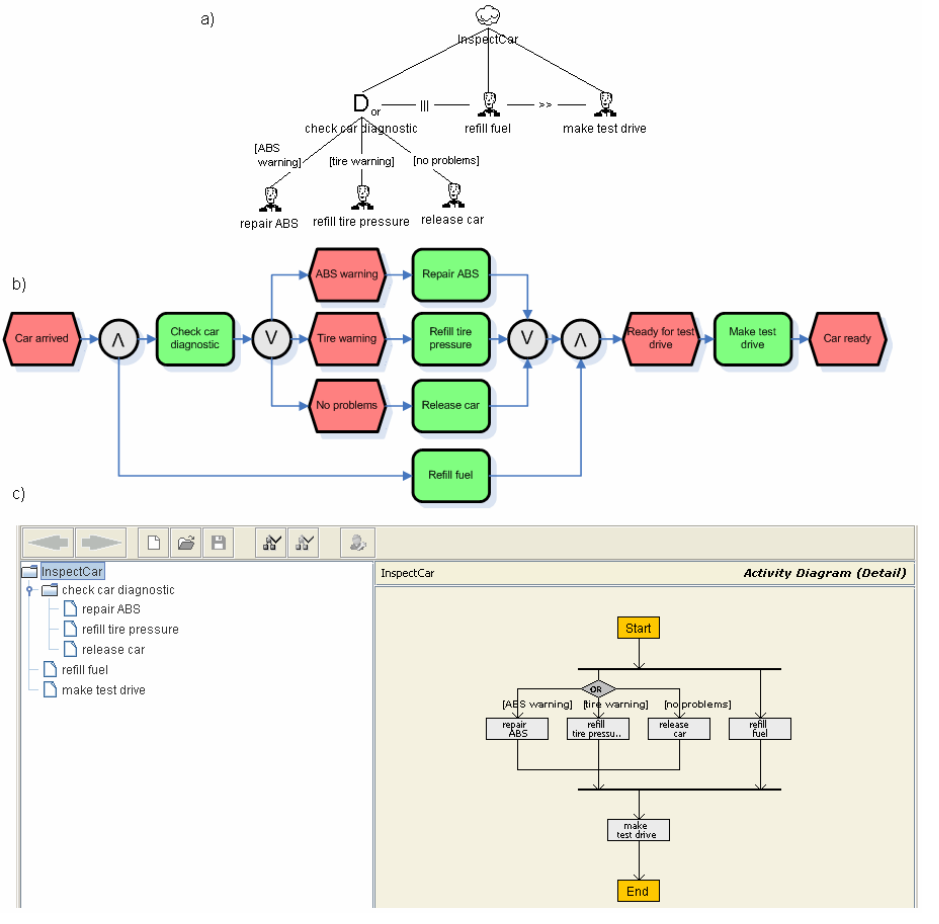


Fig. 3. The process *inspect car* displayed as a) CTT model with a decision node, b) EPC with a decision function and c) TTMS model displayed in the editor

Decisions made via an instantiation time choice cannot be decided differently at runtime within iterations. In contrast, if a decision is specified in an iteration, for every iteration cycle the decision can be made differently. In this respect, the decision modeling concept of the instantiation time choice is fundamentally different to the explicit runtime choice. Compared to the explicit choice, it is not possible to combine the TTMS choice that reacts like a deferred choice with the instantiation time concept.

The instantiation time can for example represent the time when an order is accepted by the customer service in a company. The process to fulfill the customers

order is configured for the individual clients or customers needs. In contrast, C-EPC models are used to represent reference models for different branches or companies to be configured to individual processes of a particular company at configuration time [19]. Within the concept presented here, the configuration time is deferred to the latest possible moment before runtime.

The application of the instantiation time concept with the WfMS will be described more deeply in subsection 3.1.

2.3 Task Model Development - Editor and Tool Chain

The proposed tool chain for workflow management with TTMS is pictured in Figure 4. First of all, the CTTE tool is used to model and easily animate workflows in the modeling environment similar to the model of Figure 2b). Workflow models can be simulated with explicit *start* and *finish* events as discussed in subsection 2.1 within the workflow animation. The editing functions are mature in CTTE so that new hierarchy levels can be inserted quite easily. Subtrees can be cut and pasted to other parts of the process tree. Normally a top-down approach is followed to model the task models. But with its mature editing possibilities also a buttom-up modeling approach can be followed.

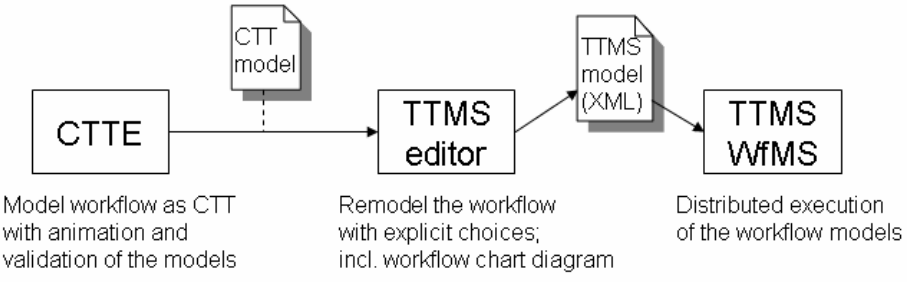


Fig. 4. Tool chain from task tree modeling in CTTE up to the deployment of the workflow models in TTMS

After developing and validating in CTTE, models are reworked in the TTMS editor. The TTMS editor is an application to model workflows. It produces XML files that are interpreted by the TTMS WfMS. The WfMC reference model describes this functionality with the interface 1 [9], so that this is implemented in TTMS. In Figure 3c) the flat activity specification of the task tree can be seen in the workflow chart. All leafs of the tree are pictured as activities and all inner nodes of the task tree are omitted. The task *InspectCar* is selected in the tree pictured at the left hand side of Figure 3c) so that all subordinated elements in the tree are considered in the workflow chart on the right hand side of that figure. The tasks in the same or higher level in the tree are not displayed. The structural character of the workflow model [11] is clearly visible in the workflow chart. One *Start* and one *End* node are pictured which encapsulate the single-entry-single-exit region [21]. The editor provides also an

aggregated view, showing only first level subtasks below the selected task pictured in Figure 5b). Thus, workflow models are pictured as flowcharts in the TTMS editor. This might improve the understanding for business process modelers compared to the CTT diagram layout. The CTT models are enriched with decision nodes in TTMS for explicit decision modeling, which is discussed in subsection 2.2. For these operators the instantiation time decision can be used instead.

The model is finally stored into an XML file using a special internal format understood by the TTMS WfMS, the last tool of the chain. In TTMS the models will be instantiated and executed. TTMS is a distributed web-based system. The workflow engine resides on a server and guides independent clients through the predefined workflows.

Please note the different object flow notations shown in Figure 4 which each express different circumstances. Dashed lines express that the CTT models created by the CTTE tool are not automatically understood by the TTMS editor. Instead the validated CTT models are to be used as blueprints for the modeler to create the according TTMS models. The second object flow in the tool chain shows that the TTMS models are stored by the editor in a common TTMS-XML format.

3 TTMS – Task Tree Based Workflow Management System

This section introduces the developed WfMS by describing the functionality and user interface of TTMS. It implements the interface 1 concept of the WfMC, being divided into an editor and a WfMS [9]. The output of the workflow editor is an XML file which the WfMS accepts and interprets. Figure 4 gives a rough conceptual overview on TTMS system architecture.

3.1 Instantiating the Process

The WfMS has two sources to instantiate workflows from. A database backend for permanently stored flows and a mechanism to manually load temporary workflows from TTMS-XML files. After a process model has been selected by either of these two ways the process model is evaluated and checked for any instantiation time operators. These operators would be evaluated first, conceptually between build time and runtime during the instantiation time [2, 22]. The process model is configured right before the beginning of the workflow of the corresponding process instance.

In Figure 5a) the process model of Figure 3 is enriched with decision node *negotiate customers needs*. An instantiation time multi-choice (OR) operator is used for that decision node. The workflow chart shown in Figure 5b) in combination with the tree of Figure 5a) shows the instantiation time multi-choice operator as *I-OR*. The workflow chart in b) shows the aggregated view of the structured workflow chart in contrast to the model view in Figure 3. The decision task *check car diagnostic* has sub activities that are collapsed into that node in Figure 3b). The color shade is a little darker to identify an aggregated node.

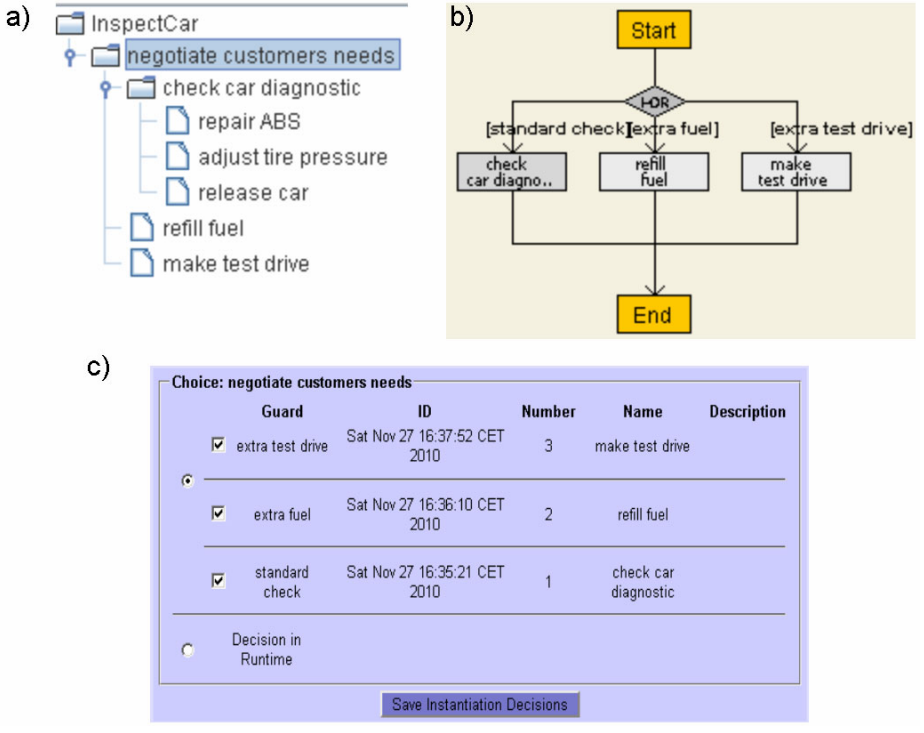


Fig. 5. a) Task tree with decision node in the TTMS editor b) workflow chart in the TTMS editor c) Execution of the instantiation time OR operator in the TTMS WfMS

The instantiation time choice operator is evaluated by the TTMS WfMS in Figure 5c) just after the user gave the command to create a process instance. The WfMS asks by executing the decision node operator *negotiate customers needs*, which services are desired by the client for the car inspection. During the execution of this operator the three services of the model of Figure 2a) are offered to the customer for selection by checkboxes. Furthermore, a possibility is given to defer the decision into the runtime. The user has to choose between these options. If she decides to defer the decision into the runtime the question *negotiate customers needs* will be asked and the related options will be offered then again. Following this possibility, different options can be selected during runtime within iterations in contrast of making the decision at instantiation time.

3.2 Managing Workflow Instances (Control Perspective)

When a workflow is instantiated its instance can be executed by the assigned user. The example of Figure 6 pictures the control perspective of the TTMS WfMS. Several views can be seen. On the left hand side currently instantiated processes are listed

in the tree just under the root node *Projects*. In case of Figure 6 two processes are active. *InspectCar* is a process with instantiation time choice operator *negotiate customers needs* specified like in Figure 5a) and b) and after evaluating the instantiation time operator similar to Figure 5c).

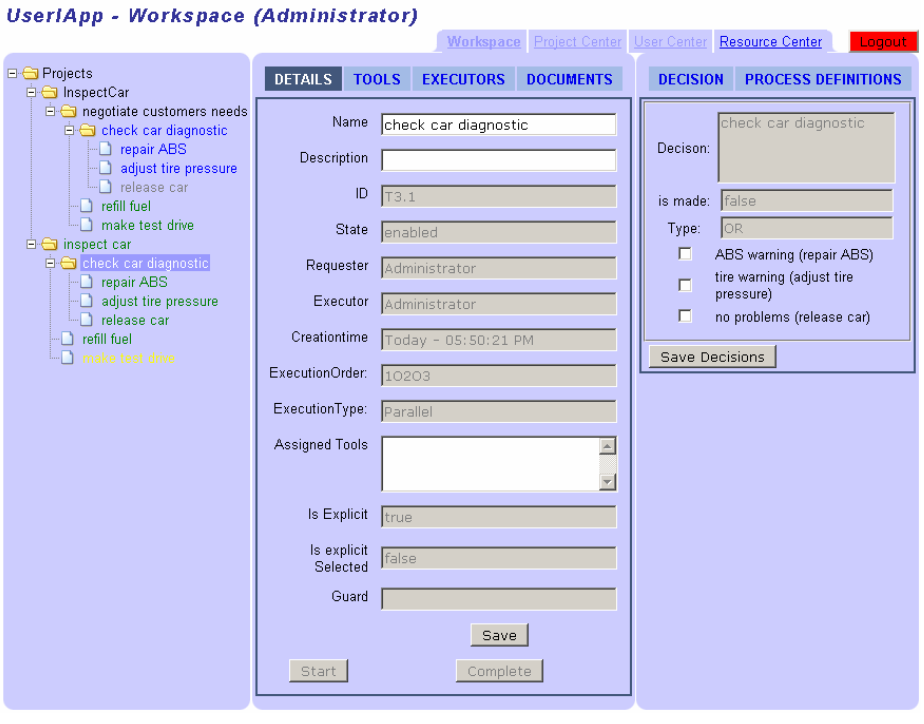


Fig. 6. User application of the TTMS WfMS

The second process instance is the process *inspect car* which is a process without instantiation time choice operator as specified in Figure 3. The decision node *check car diagnostic* is selected in the control view and on the right hand side the choice view is opened. A multi-choice operator is interpreted so that the user can select multiple cases by selecting the relevant checkboxes connected to the guards. In brackets the root node of the related subtrees are indicated. The guards specify the criteria to select the corresponding subtree and the root node specifies the work to be done.

Concerning the inspect car example of Figure 6, the user is probably a technical employee or rather mechanic of a car garage who has to do the *check car diagnostic* decision activity. To perform the real work and to execute the decision node is that she has to work with the car diagnostic device and carry over the corresponding displayed information to the WfMS. For example if the diagnostic device is showing an *ABS warning* the corresponding checkbox should be selected. After clicking on *save decisions* under the checkboxes pictured in Figure 6 the decision activity is finished and selected subtrees are enabled and the others skipped.

In the middle of the window pictured in Figure 6 the *DETAIL* view is selected. The details of the selected activity are used to view. Important to note are the two buttons *start* and *complete* near the bottom of that window. With them an employee can start and stop activities similar to the interaction in other WfMS like YAWL. This is the user interaction that was simulated with CTTE in Figure 2b).

The representation of workflows as trees shown on the left hand side of Figure 6 might help the executing person to understand the workflows better than having a worklist like in other WfMS. The context goals are given with the inner nodes of the tree model. The state *executed*, *enabled*, *skipped* or *waiting* of the tasks are shown with a particular color in the WfMS.

On top of Figure 6 some links called *TOOLS*, *EXECUTORS* and *DOCUMENTS* can be seen. Following them, the user can go to the resource, role and data view of the selected activity that are also implemented in the TTMS system. The management of the users, roles and resources are implemented independently to the task tree concept. In models like CTT these aspects are not explicit features of the language in contrast to other workflow languages like EPCs, UML activity diagrams or BPMN.

4 Related Work

Some Workflow Management Systems are implemented in combination with several languages that are interpreted by the corresponding WfMS. Yet another workflow language (YAWL) [24] is such a language based on the Petri nets theory to model workflows. An editor can create the models and a web-based WfMS is used to instantiate and manage these models. ADEPT is a flexible workflow management system [5] in which workflow models can be adapted during the runtime after the “flexible by change” principle. YAWL as well as ADEPT are not used to handle hierarchical workflow models in such an easy and integrated way like TTMS does. Nevertheless ADEPT workflow models are also structured and sound by construction so that transformations to TTMS seems to be possible.

A WfMS that uses hierarchical workflow specification is implemented with WASA [25]. In contrast to the TTMS system, data flow aspects can be modeled with dashed lines in the WASA models. But they seams to be unstructured, more confusing and less readable compared to the task models like CTT or TTMS as structured workflow models. TTMS is the only WfMS of the listed ones that implements the EPC related explicit choice and instantiation time configuration mechanism.

Hierarchical workflow models that are rather used for business process modeling than for workflow management exist with function trees in the ARIS method [20] or hierarchical supply and value chains [18]. Furthermore, there exist papers about goal oriented process design with EPCs together with hierarchical goal models [16]. Process structure trees [21] are used to express process models as trees. In contrast, HTA [6], GOMS [6] or CTT [17] are hierarchical process modeling languages for activity modeling in the HCI context. They are used for example for usability evaluation [6, 17] or model-based user interface development [12, 6, pp.135-155]. In these domains action modeling of the user is essential and is done in a human centered way. The work presented in this paper tries to connect the field of workflow modeling and management with task modeling field by implementing a WfMS that uses task trees as models for workflow execution.

5 Summary

This paper has introduced an approach to model workflows in a hierarchical, structured way by using task trees. They are frequently used in the HCI community to specify tasks and user actions in a goal or rather problem oriented way. The models are interpreted in a WfMS that also presents the models in a tree-like notation. The hierarchical workflow model inherently describes the context goals of tasks in contrast to a flat workflow representation. Thus, different abstraction levels in one model might help end-users to understand the models better during the development at design time as well as at runtime.

The models are structured and for that reason better readable and understandable [14]. A mature tool CTTE was proposed to be used in the tool chain within the TTMS approach presented in this paper.

Parts of the WfMC reference model are implemented in the TTMS. The models can be used to manage the workflows in a company or organization. Furthermore, TTMS provides a system to execute and thus validate the cooperative parts of the task models in a real distributed environment in which end-users can work guided by the WfMS in their usual working environment. The modeling tool CTTE provides only a validation of the dynamic properties of the models by executing them on a stationary computer in contrast to TTMS.

Differences and similarities of task modeling to business process and workflow modeling were discussed in this paper. In this context it is stated that task modeling normally starts with a human goal and goes beyond business process modeling level to a fine granular user action level. In contrast, task trees used in a WfMS starts with a task or goal of the company and stops at a level above fine granular user actions.

Furthermore, decision modeling in task trees is compared to decision modeling in EPCs. Explicit choice operators are added with decision nodes in task trees. The multi-choice (OR) operator is integrated and implemented in the course of that. Moreover, an instantiation time concept is implemented. Process models can be configured to the customers needs by the WfMS just before execution of the process model. This is even true for parts of a model.

References

1. Balzert, H.: Lehrbuch der Software-Technik: Basiskonzepte und Requirements Engineering. Spektrum, Heidelberg (2009)
2. Brüning, J., Forbrig, P.: Methoden zur adaptiven Anpassung von EPKs an individuelle Anforderungen vor der Abarbeitung. In Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (EPK 2008), Saarbrücken, CEUR-WS, vol. 420 (2008)
3. Brüning, J., Forbrig, P.: Modellierung von Entscheidungen und Interpretation von Entscheidungsoperatoren in einem WfMS. In: Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (EPK 2009), Berlin, CEUR-WS, vol. 554 (2009)
4. Brüning, J., Dittmar, A., Forbrig, P., Reichart, D.: Getting SW Engineers on Board: Task Modelling with Activity Diagrams. In: Gulliksen, J., Harning, M.B., van der Veer, G.C., Wesson, J. (eds.) EIS 2007. LNCS, vol. 4940, Springer, Heidelberg (2008)
5. Dadam, P., Reichert, M.: The ADEPT project: A decade of research and development for robust and flexible process support - challenges and achievements. Computer Science - Research and Development 22, 81–97 (2009)

6. Diaper, D., Stanton, N.: The Handbook of Task Analysis for Human-Computer Interaction. Lawrence Erlbaum Assoc. Inc., Mahwah (2003)
7. Eriksson, H.-E., Penker, M.: Business Modeling With UML: Business Patterns at Work. Wiley, Chichester (2000)
8. Gabriel, M., Ferreira, V., Ferreira, D.R.: Understanding Spaghetti Models with Sequence Clustering for ProM. In: Business Process Management Workshops (BPM 2009 International Workshops), Ulm. LNBIP, vol. 43 (2009)
9. Hollingsworth, D.: The Workflow Reference Model. Tech. Rep. Document Number TC00-1003, Workflow Management Coalition
10. Keller, G., Nüttgens, M., Scheer, A.-W.: Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK). In: Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi), Heft 89, Universität des Saarlandes (January 1992)
11. Kiepuszewski, B., ter Hofstede, A.H.M., Bussler, C.J.: On structured workflow modelling. In: Wangler, B., Bergman, L.D. (eds.) CAiSE 2000. LNCS, vol. 1789, p. 431. Springer, Heidelberg (2000)
12. Luyten, K.: Dynamic User Interface Generation for Mobile and Embedded Systems with Model-Based User Interface Development, PhD Thesis in University Limburg (2004)
13. La Rosa, M., Gottschalk, F., Dumas, M., van der Aalst, W.: Linking Domain Models and Process Models for Reference Model Configuration. In: Proceedings of the International Conference on Business Process Management, BPM 2007 (2007)
14. Laue, R., Mendling, L.: The Impact of Structuredness on Error Probability of Process Models. In: Information Systems and e-Business Technologies 2nd International United Information Systems Conference UNISCON 2008, Klagenfurt. LNBIP, vol. 5 (2008)
15. Mori, G., Paterno, F., Santoro, C.: CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. IEEE Transactions on Software Engineering, 797–813 (2002)
16. Neiger, D., Churilov, L.: Goal-Oriented Decomposition of Event-Driven Process Chains with Value focused Thinking. In: 14th Australasian Conference on Information Systems, Perth
17. Paterno, F.: Model-Based Design and Evaluation of Interactive Applications. Springer, Heidelberg (2000)
18. Poluha, R.G.: Application of the SCOR Model in Supply Chain Management. Youngstown, New York (2007)
19. Rosemann, M., van der Aalst, W.: A configurable reference modelling language. Information Systems 32(1), S.1–S.23 (2007)
20. Scheer, A.-W.: ARIS: Business Process Modeling. Springer, Heidelberg (1999)
21. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 100–115. Springer, Heidelberg (2008)
22. van der Aalst, W., Pesic, M., Schonenberg, H.: Declarative Workflows Balancing Between Flexibility and Support. Computer Science - Research and Development 23(2), 99–113 (2009)
23. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow Patterns. Distributed and Parallel Databases 14(3), 5–51 (2003)
24. van der Aalst, W., ter Hofstede, A.: YAWL – Yet Another Workflow Language (Revised version). QUT Technical Report, FIT-TR-2003-04, Queensland University of Technology, Brisbane (2003)
25. Weske, M.: Workflow Management Systems: Formal Foundation, Conceptual Design, Implementation Aspects. Habilitation Thesis, University of Münster (2000)

A Modeling Paradigm for Integrating Processes and Data at the Micro Level

Vera Künzle and Manfred Reichert

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

Abstract. Despite the widespread adoption of BPM, there exist many business processes not adequately supported by existing BPM technology. In previous work we reported on the properties of these processes. As a major insight we learned that, in accordance to the data model comprising object types and object relations, the modeling and execution of processes can be based on two levels of granularity: object behavior and object interactions. This paper focuses on micro processes capturing object behavior and constituting a fundamental pillar of our framework for object-aware process management. Our approach applies the well established concept of modeling object behavior in terms of states and state transitions. Opposed to existing work, we establish a mapping between attribute values and objects states to ensure compliance between them. Finally, we provide a well-defined operational semantics enabling the automatic and dynamic generation of most end-user components at run-time (e.g., overview tables and user forms).

Keywords: Object-aware Processes, Data-driven Process Execution.

1 Introduction

Process Management Systems (PrMS) enable the modeling, execution and monitoring of business processes [1]. Despite their widespread adoption, there exist many knowledge-intensive processes which cannot be "straight-jacketed into activities" [2,3]. Prescribing an activity-centred process model for them would lead to a "contradiction between the way processes can be modeled and the preferred work practice" [4]. Moreover, PrMS do not provide integrated access to application data. In particular, end-users cannot access application data at any point in time (assuming proper authorization). In this context, overview tables (e.g., data reports) and user forms constitute important components. The latter provide (data) input fields (e.g., textfields, checkboxes) for reading and writing selected attribute values of object instances. Further, many activities of a process model are implemented as forms. As known from practice, however, implementing the logic of these forms and other user components causes high implementation efforts.

In the PHILharmonicFlows[1] project, we are developing concepts, methods and tools for realizing object- and process-aware information systems [5]. In

[1] Process, Humans and Information Linkage for harmonic Business Flows.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 201–215, 2011.

particular, we are targeting at a flexible integration of business data, business processes, business functions, and users to overcome limitations known from activity-centered PrMS. In addition, we aim at the automatic generation of end-user components; e.g., tables giving an overview on a collection of object instances and form-based activities (including their internal logic). This way, not only generic process support, but also generated application functionality shall be provided.

This paper introduces a fundamental pillar of our PHILharmonicFlows framework by introducing an advanced paradigm for the modeling and run-time support of object behavior, i.e., the processing of individual object instances. The latter provides the foundation of object-aware processes involving multiple object instances of the same and of different object type. Like existing work considering object behavior during process execution [6,7,8,9,10,11,12,13] our approach applies the well established concept of modeling object behavior in terms of states and state transitions. Opposed to existing approaches (e.g., case handling), however, PHILharmonicFlows enables a mapping between attribute values and objects states and therefore ensures compliance between them. In addition, integrated access to application data is provided. Here, not only generic process support, but also generated application functionality is provided. Finally, the presented execution paradigm combines data-driven process execution with activity-oriented aspects.

Section 2 illustrates the research methodology we applied and discusses major requirements for the modeling and run-time support of object behavior. An overview of our framework is given in Section 3. We introduce its underlying data model in Section 4 and the modeling of object behavior in Section 5. Section 6 deals with authorization issues targeting at the automatic generation of (form-based) activities at run-time. The corresponding execution paradigm is discussed in Section 7. Section 8 investigates related work and Section 9 closes with a summary and outlook.

2 Research Methodology

To better understand the characteristics of processes that are well supported by existing technology and those handled insufficiently, we analyzed many processes from domains like healthcare, human resource management, and automotive engineering [14,15,5]. As fundamental insight we gained from these studies that many processes require *object-awareness*; i.e., full integration of application data consisting of object types, object attributes, and object relations. In previous work we identified the properties of these processes [5,16] and discussed challenges to be tackled for integrating processes, data and users [14,15]. A major finding was that there are strong relationships between process support and data management. In accordance to the data model comprising *object types* and *object relations*, therefore, the modeling and execution of processes is based on two levels of granularity as well: *object behavior* and *object interactions*. Regarding object behavior the following properties are significant: For each *object instance* a corresponding *process instance* should exist controlling its processing.

Object attribute values reflect the progress of this process instance. While certain attribute values can be optionally assigned, others are mandatorily required in order to reach a particular process goal. For this purpose, *mandatory activities* need to be enabled and assigned to responsible users if required information is missing. In addition, *optional activities* for reading and writing attribute values at any point in time should be supported. Generally, one has to ensure that *object state* and *process state* are compliant with each other. Further, it should be possible to enter data at the moment it becomes available (i.e., using optional activities). In particular, users should be allowed to enter data up-front; i.e., before the corresponding mandatory activity becomes enabled. For this purpose, it should be possible to drive process execution based on data and to dynamically react upon attribute value changes. Mandatory activities no longer needed (due to an up-front data entry) should then be automatically skipped. Moreover, users should be enabled to re-execute activities until they explicitly commit their completion. Generally, different ways for reaching a process goal exist. In our context, this selection might be also based on explicit user decisions. When filling in forms, certain attribute values might become mandatory on-the-fly; i.e., whether or not an object attribute is mandatory may depend on other object attribute values. It should therefore be possible to manage the internal flow of control within particular activities (e.g. user forms) as well. Finally, such integration of process and data necessitates advanced concepts for user integration; i.e., process authorization must be compliant with data authorization and vice versa. While certain users must execute an activity mandatorily in the context of a particular object instance, others may be authorized to optionally execute this activity.

We have already shown that only limited support for these properties is provided by existing imperative, declarative, and data-driven process support paradigms [16]. To ensure the relevance, completeness and generalizability of the identified properties we performed a literature study concerning extensions of the basic paradigms (i.e., imperative, declaratives and data-driven ones) [5]. Finally, we are currently developing a proof-of-concept prototype of our framework.

3 PHILharmonicFlows Framework

This section gives a short overview of our PHILharmonicFlows framework which enforces a well-defined modeling methodology governing the definition of processes at different levels of granularity and being based on a well-defined formal operational semantics. More precisely, the framework differentiates between *micro* and *macro processes* in order to capture both *object behavior* and *object interactions*. As a prerequisite, object types and their relations need to be captured in a data model. Following this, for each *object type* a *micro process type* has to be specified. The latter defines the behavior of corresponding object instances and consists of a set of *states* and the *transitions* between them. Each state is associated with a set of *object type attributes*. At runtime, a micro process instance being in a particular state may only proceed if specific values are

assigned to the attributes associated with this state; i.e., a *data-driven process execution* is applied. *Optional access* to data, in turn, is enabled asynchronously to process execution and is based on permissions for creating and deleting object instances as well as for reading/writing their attributes. The latter must take the current progress of the corresponding micro process instance into account. For this, PHILharmonicFlows maintains a comprehensive *authorization table* assigning data permissions to user roles depending on the different states of the micro process type. Taking the relations between the object instances of the overall *data structure* into account, the corresponding micro process instances additionally form a complex *process structure*; i.e., their execution needs to be coordinated according to the given data structure. In PHILharmonicFlows this can be realized by means of macro processes. Such a *macro process* consists of *macro steps* as well as *macro transitions* between them. Opposed to micro steps that relate to single attributes of a particular object type, a macro step refers to an entire object type and a particular state. For each macro transition, a corresponding synchronization component must then be specified. This way, PHILharmonicFlows is able to hide the complexity of large process structures from modelers as well as from end-users. The synchronization components enable the coordination of interactions between the object instances of the same as well as of different object types. Opposed to existing approaches, it is possible to additionally consider the cardinalities between object instances. In particular, whether or not a particular object instance should be (mandatorily or optionally) created depends on the relation cardinalities and on synchronization components specified within the macro process.

4 Modeling Data

As opposed to existing approaches, in which activities and their execution constraints (e.g., precedence relations) are explicitly specified, PHILharmonicFlows allows defining processes by taking *object types* as well as *object interactions* into account. For this purpose, the proper integration of data constitutes a fundamental requirement. Regarding existing process support approaches, however, the data and process perspectives are mostly integrated in one and the same model leading to complex and overloaded models being difficult to maintain. PHILharmonicFlows, in turn, supports the definition of data and processes in separate, but well integrated models. Thus, it retains the well established principle of separating concerns [17].

Due to the widespread use of the relational data model, PHILharmonicFlows is based on relational concepts as well. In this paper, we restrict the data perspective to object types and object attributes (see [5] for our basic idea on how to treat object relations). As example consider review processes for job applications as known from the human resource area. In this real world example, which we simplified for the sake of clarity, `reviews` are used to evaluate applications and are provided by employees from functional divisions. Based on the results of the `reviews` the personnel officer from the human resource department decides which applicant may get the offered job. For this purpose, a `review` object

type may comprise attribute types like `issue date`, `proposal` (e.g., to invite or reject the applicant), `appraisal`, `remark`, `comment`, `reason`, `consideration` (indicating whether the result of the review was used to initiate further actions), and `appointment` (suggested date for interview) (cf. Fig. 1a). Attribute types are represented by atomic data elements with a specific data type (i.e., integer, decimal, string, boolean, date). Arrays and sets, in turn, are captured as relating object types (but are out of the scope of this paper). Finally, an object type comprises a set of attribute types defining its properties.

Let *Identifiers* be the set of all valid identifiers over a given alphabet.

Definition 1. *An **attribute type** is a tuple attrType = (name, type) where*

- *name* $\in$ *Identifiers is an identifier.*
- *type* $\in$ $\{INTEGER, DECIMAL, STRING, BOOLEAN, DATE\}$ *is a basic data type.*

Further, AttrTypes denotes the set of all definable attribute types and AttrValues the set of all possible attribute values given the above set of data types.

Definition 2. *An **object type** is a tuple oType = (name, AttrTypeSet) where*

- *name* $\in$ *Identifiers is an identifier.*
- *AttrTypeSet* $\subset$ *AttrTypes is a finite set of attribute types.*

Further, $\forall$ $attrType_1$, $attrType_2 \in AttrTypeSet$:
$attrType_1.name = attrType_2.name \Rightarrow attrType_1 \equiv attrType_2$.
Finally, OTypes corresponds to the set of all definable object types.

5 Modeling Object Behavior

Our PHILharmonicFlows framework enforces a well-defined modeling methodology governing the definition of processes at different levels of granularity. More precisely, the framework differentiates between *micro* and *macro processes* in order to capture both *object behavior* and *object interactions*. This section introduces our modeling approach for micro process types capturing object behavior.

For each *object type* one specific *micro process type* comprising a number of *micro step types* has to be defined. Each micro step type, in turn, is associated with an attribute type and describes an elementary action (e.g. writing the object attribute). By connecting micro step types using *micro transition types*, we obtain their default execution order. Further, *state types* can be used to realize mandatory activities comprising a subset of the micro step types; i.e., they are used to coordinate actions between different users. Thereby, the micro step types belonging to the same state type and their relations reflect the internal logic of an activity, whereas state types are used to coordinate the execution of several activities among different user roles.

Fig. 1b shows the micro process type describing the behavior of the `review` object type (cf. Fig. 1a). Each `review` must be created by a personnel officer and then be filled out by an employee. The latter can either refuse the `review` request or fill out the corresponding `review` form. In the latter case, the personnel officer has to evaluate the feedback provided by the employee. For this purpose, our example comprises four state types. These represent mandatory activities involving two roles (cf. Fig. 1b). Further, each micro process type includes at least one *end state*; i.e., an object state in which no further actions are mandatorily required. Our `review` micro process type has two end states, namely `evaluated` and `closed` (cf. Fig. 1b).

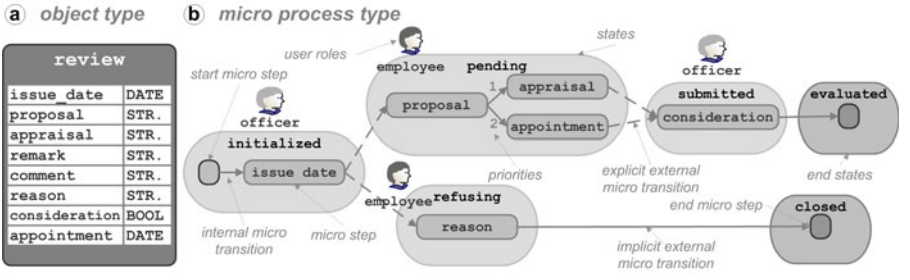


Fig. 1. Review object type and micro process type

We first discuss how to specify the internal logic of a *mandatory activity* as captured by a state and its corresponding micro steps. For each state type, we define a set of corresponding actions. More precisely, each *state type* comprises several *micro step types*. Each of them may represent a mandatory write access to a particular object attribute. Note that single micro step types do not represent activities, but solely refer to one atomic action (e.g., editing an input field within a form). As we will show later, we do not need to explicitly model activities (and corresponding forms), but can *automatically generate* them. Here, also optional input fields as well as read-only data fields are integrated based on a sophisticated authorization table (cf. Fig. 3).

Each micro step type may refer to a specific attribute type. For reaching a micro step during run-time, a value for its corresponding attribute is mandatorily required. As illustrated in Fig. 1b, when initializing a `review` (i.e., state `initialized` is activated), a personnel officer must specify an `issue date`. Besides, state types do not require further actions. As example consider end states and states which only require an explicit commit by a responsible user (e.g., enabling mandatory reading). Respective state types only comprise micro step types not referring to any attribute type.

When executing mandatory activities, users should be guided in setting required attribute values (e.g., by highlighting respective input fields in a form). Regarding state `pending` in Fig. 1b, for example, after setting the value of attribute `proposal`, either the value of attribute `appraisal` or attribute `appointment` is required next. To capture such logic for the setting of object attributes, their

micro step types can be linked using *micro transition types*. Based on them, we can define the *internal logic* of a mandatory activity; e.g., the default order in which the input fields of the corresponding form shall be edited. If a micro step type (e.g., `proposal`) contains more than one outgoing micro transition type (i.e., an alternative processing), we ensure that only one of them is fired at runtime; i.e., always one micro step (and thereby also one state) can be reached.[2] Regarding our example, the values of attributes `proposal` and `appointment` are usually set when enabling respective micro steps. However, an employee may set these values early on. In such case, the micro steps corresponding to these object attributes will be immediately completed when they are reached. If values for both `appointment` and `appraisal` are available, we have to ensure that only one micro step is activated. For this purpose, different *priorities* as illustrated in Fig. 1b can be assigned to micro transition types. Regarding our example, the micro step referring to attribute `appraisal` would be reached because the corresponding incoming transition has a higher priority as the one of the micro step referring to attribute `appointment`. If only a value for attribute `appointment` was available, in turn, its corresponding micro step would be activated.

Definition 3. *An **micro process type** is a tuple micProcType = (oType, MicStepTypeSet, MicTransTypeSet) where*

- *oType = (name, AttrTypeSet) ∈ OTypes is the object type whose behavior is described by micProcType.*
- *MicStepTypeSet is a finite set of micro step types with micStepType = (name, attrType) ∈ MicStepTypeSet having the following meaning:*
  - *name ∈ Identifiers is an identifier.*
  - *attrType ∈ AttrTypeSet ∪ {NULL} is an attribute type or undefined.*
- *MicTransTypeSet ⊂ MicStepTypeSet × MicStepTypeSet × ℕ is a finite set of micro transition types with micTransType = (source, target, priority) ∈ MicTransTypeSet having the following meaning:*
  - *source ∈ MicStepTypeSet is the source micro step type of micTransType.*
  - *target ∈ MicStepTypeSet is the target micro step type of micTransType.*
  - *priority ∈ ℕ is the priority of micTransType.*

MicProcTypes denotes the set of all definable micro process types.

PHILharmonicFlows provides support for backward jumps within micro processes as well (resetting attribute values where required). However, due to lack of space we only consider acyclic micro process types here. Each micro process type contains exactly one start micro step type which does not refer to any attribute type and has no incoming transitions (cf. Def. 4a). Further, a micro

[2] Though an object instance is always in exactly one processing state, this does not prohibit parallel execution. During the execution of an activity, parallel processing of disjoint sets of mandatory as well as optional object attributes is always possible. In addition, different users may concurrently process forms corresponding to the same object instance. In this context, known mechanisms for synchronizing concurrent data access can be applied.

process type must comprise at least one end micro step type which does not refer to an attribute type and has no outgoing micro transition type (cf. Def. 4b). All other micro step types, in turn, must have at least one incoming (cf. Def. 4c) and at least one outgoing micro transition type (cf. Def. 4d). To ensure this we introduce functions for structural analysis of *micProcType = (oType, MicStepTypeSet, MicTransTypeSet) ∈ MicProcTypeSet*. In particular, *intrans: MicStepTypeSet* $\mapsto \mathbb{N}_0$ (*outrans: MicStepTypeSet* $\mapsto \mathbb{N}_0$) determines for each micro step type the number of its incoming (outgoing) micro transition types. To ensure that only one micro step is activated at any point during run-time, micro transition types having the same source micro step type must be associated with different priorities (cf. Def. 4e).

Definition 4. *Let micProcType = (oType, MicStepTypeSet, MicTransTypeSet) ∈ MicProcTypes be an acyclic micro process type referring to an object type oType = (name, AttrTypeSet) ∈ OTypes. Then:*

a) $\nexists$ $startMicStepType_{micProcType}$ *= (name, NULL) ∈ MicStepTypeSet with intrans(*$startMicStepType_{micProcType}$*) = 0; i.e., there exists exactly one start micro step type.*

b) $|\ EndMicStepTypes_{micProcType}\ | \geq 1$ *with* $EndMicStepTypes_{micProcType}$ *:= {micStepType = (name, NULL) ∈ MicStepTypeSet | outtrans(micStepType) = 0}; i.e., there exists at least one end micro step type.*

c) $\forall$ *micStepType ∈ MicStepTypeSet -* $\{startMicStepType_{micProcType}\}$*: intrans(micStepType) ≠ 0.*

d) $\forall$ *micStepType ∈ MicStepTypeSet -* $EndMicStepTypes_{micProcType}$*: outtrans(micStepType) ≠ 0.*

e) $\forall$ $transType_i \in MicTransTypeSet$*, i=1,2:* $transType_1 \neq transType_2 \wedge transType_1.source = transType_2.source \Rightarrow transType_1.priority \neq transType_2.priority$.

Several micro step types can be aggregated to a particular state type:

Definition 5. *Let micProcType = (oType, MicStepTypeSet, MicTransTypeSet) ∈ MicProcTypes be an acyclic micro process type. A* ***state type*** *of a micro process type micProcType is a tuple stateType = (name, sMicStepTypeSet) where*

- *name ∈ Identifiers is an identifier.*
- *sMicStepTypeSet ⊆ MicStepTypeSet is a finite set of micro step types.*

$StateTypes_{micProcType}$ *is a finite set of state types defined on micProcType.*

In Fig. 1b, `pending` is an example of a state type comprising the micro step types `proposal`, `appraisal` and `appointment`. Generally, different state types have disjoint sets of micro step types (cf. Def. 6a) and each micro step type must belong to exactly one state type (cf. Def. 6b). In addition, each *end* micro step type must belong to a state type comprising no other micro step types (cf. Def. 6c). Further, the micro step types belonging to the same state type must be connected with each other (cf. Def. 6d).

Definition 6. *Let micProcType = (oType, MicStepTypeSet, MicTransTypeSet) ∈ MicProcTypes be an acyclic micro process type and let $stateType_i \in StateTypes_{micProcType}$, i = 1,2 be two state types. Then:*

a) $stateType_1 \equiv stateType_2 \Rightarrow$ $stateType_1.sMicStepTypeSet \cap stateType_2.sMicStepTypeSet = \emptyset$

b) $\forall$ *micStepType* $\in$ *MicStepTypeSet:* $\nexists$ *stateType* $\in StateTypes_{micProcType}$*:* *micStepType* $\in$ *stateType.sMicStepTypeSet*

c) $\forall$ *micStepType* $\in EndMicStepTypes_{micProcType}$*:* $\nexists$ *stateType* $\in StateTypes_{micProcType}$*:* *micStepType* $\in$ *stateType.sMicStepTypeSet* $\wedge$ $|$ *stateType.sMicStepTypeSet* $|$ = 1.

d) $\forall$ *stateType* $\in StateTypes_{micProcType}$*:* $micStepType_i \in$ *stateType.sMicStepTypeSet, i = 1,2* $\wedge$ $micStepType_2$ *is a successor of* $micStepType_1 \Rightarrow$ *all micro step types on the path from* $micStepType_1$ *to* $micStepType_2$ *belong to stateType.sMicStepTypeSet as well.*

We further denote *micro transition types* that connect *micro step types* belonging to different *state types* as *external micro transition types.*
Formally: *isexternal: MicTransTypes* $\mapsto$ *BOOLEAN* with:

$$\text{isexternal(mtt)}:=\begin{cases} TRUE, & \exists stateType_i \in StateTypes_{micProcType}, i=1,2, \\ & \text{with } stateType_1 \neq stateType_2 \\ & \wedge mtt.source \in stateType_1.sMicStepTypeSet \\ & \wedge mtt.target \in stateType_2.sMicStepTypeSet \\ FALSE, & \text{else} \end{cases}$$

As example consider the micro transition type connecting micro step type `consideration` and the one belonging to state `evaluated` in Fig. 1b. At run-time, the firing of an external micro transition triggers a new micro state; i.e., the data-driven execution paradigm is also applied for activating subsequent states. For example, a `review` reaches state `evaluated` as soon as the responsible personnel officer has assigned the value of attribute `consideration` (cf. Fig. 1b). Opposed to a purely data-driven activation, however, some scenarios may require that a responsible user explicitly commits the completion of an activity he has worked on. As example consider state `pending`. An employee may re-execute the activity of filling in the `review` form until he explicitly commits to submit the review back to the personnel officer. To capture this in a micro process type we flag external micro transition types either as implicit or explicit:
explicit: MicTransType $\mapsto$ *BOOLEAN* defines whether a particular micro transition type *micTransType* (with *isexternal(micTransType = TRUE)*) is marked as explicit. As illustrated in Fig. 2a, to ensure that only one state of a micro process instance is activated during run-time, external micro transition types having the same micro step type as source must be defined as explicit ones (cf. Def. 7a). Certain scenarios require explicit user decisions. For example, after a personnel officer has initiated a `review`, the responsible employee may decide whether to fill in the `review` or to refuse it. In particular, a user decision is required if a micro step has more than one outgoing external, explicit micro transition. In this case,

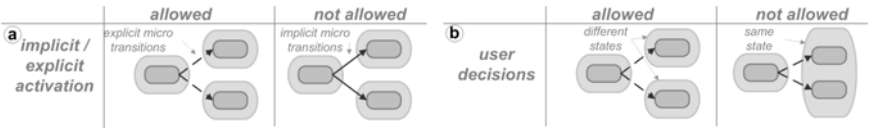


Fig. 2. Structural correctness of external transition types

the responsible user has to decide which subsequent state shall be activated. To ensure this, we have to ensure that the target micro step types of explicit external micro transition types having the same source belong to different state types (cf. Fig. 2b and Def. 7b).

Definition 7. *Let micProcType = (oType, MicStepTypeSet, MicTransTypeSet) ∈ MicProcTypes be an acyclic micro process type. Then:*

$\forall\ micTransType_i \in MicTransTypeSet,\ i{=}1,2$ *with* $micTransType_1 \neq micTransType_2 \wedge micTransType_1.source = micTransType_2.source \wedge isexternal(micTransType_i) = TRUE,\ i{=}1,2$
Then:

a) $explicit(micTransType_i) = TRUE,\ i{=}1,2$
b) $\exists\ stateType_i \in StateTypes_{micProcType},\ i{=}1,2$ *with* $stateType_1 \neq stateType_2 \wedge micTransType_i.target \in sMicStepTypeSet_i,\ i{=}1,2$

6 Data and Process Authorization

Generally, we associate state types and explicit micro transition types with user roles in order to be able to determine actors being responsible for mandatory activities, branching decisions, and commitments during run-time. In addition, it must be possible that different users (i.e., roles) may have different access rights on object attributes in a particular micro state. To achieve this, PHILharmonicFlows automatically generates a specific *authorization table* in accordance to the defined micro process type. Based on authorization tables one can define which user role may *read / write* which object attributes in the different states of a micro process (cf. Fig. 3). To ensure proper authorization, each user role assigned to a state type automatically obtains the permissions required for writing the object attributes to which the micro step types of this state type refer (see the shaded boxes in Fig. 3). The generated authorization table may be adjusted by assigning additional optional permissions allowing for the execution of optional activities. Generally, this allows users not being involved in the execution of mandatory activities to own permissions for reading or writing object attributes; e.g., a manager may read or write selected object attributes within state `submitted`. Generally, not every user being allowed to write required attribute values in a particular state should be forced to also execute the corresponding mandatory activity. To be able to differentiate between user assignment (i.e., activities a user has to do) and authorization (i.e., activities a user may do) we further distinguish between *mandatory* and *optional permissions* in

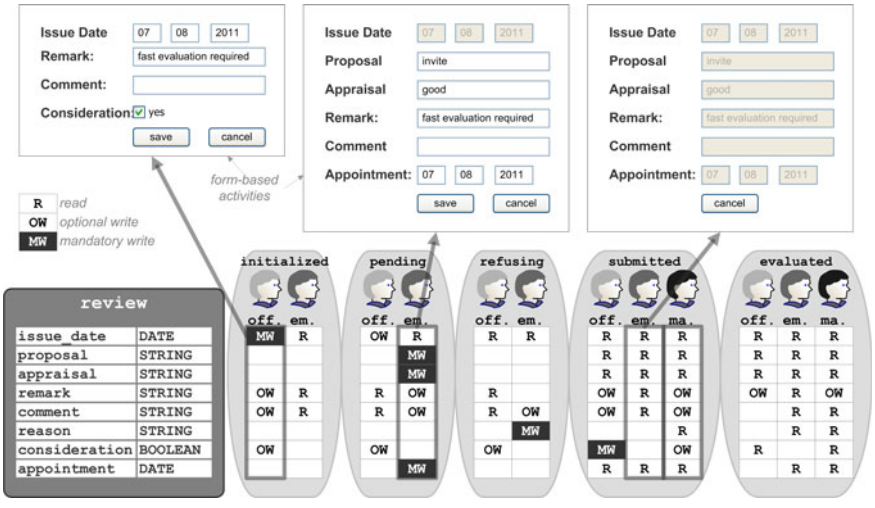


Fig. 3. Authorization table and generation of form-based activities

respect to writing object attributes. Only for users with mandatory permissions, a mandatory activity is assigned to their worklist.

7 Execution of Micro Processes

Our approach is based on a well-defined formal semantics. In particular, this enables us to automatically generate most end-user components of the run-time environment; e.g., tables giving an overview on object instances and form-based activities. Regarding the latter, the presented authorization table provides the basis for automatically generating user-specific activity forms (cf. Fig. 3); i.e., each user owing respective read and write permissions in a certain (micro process) state may execute a corresponding form-based activity. The *processing state* of an individual micro process instance is defined by the current marking of its states, micro steps, and micro transitions (cf. Fig. 4).

Instantiation. In the following, we refer to our example to demonstrate how a micro process is executed: First of all, when creating a new `reviews` object instance, a corresponding micro process instance is automatically generated and initialized. Thereby, the start micro step is marked as CONFIRMED and the state to which it belongs is marked as ACTIVATED. All other states, in turn, are initially set to WAITING. Further, the outgoing micro transition of the start step is marked as READY, whereas all other micro transitions are initially marked as WAITING. In our example, the incoming internal micro transition of micro step `issue date` is marked as READY. This, in turn, leads to marking ENABLED of the target micro step of this micro transition. Then, for this micro step a value of its corresponding attribute has to be assigned. All other micro steps belonging to the start state (state `initialized` in our example), in turn,

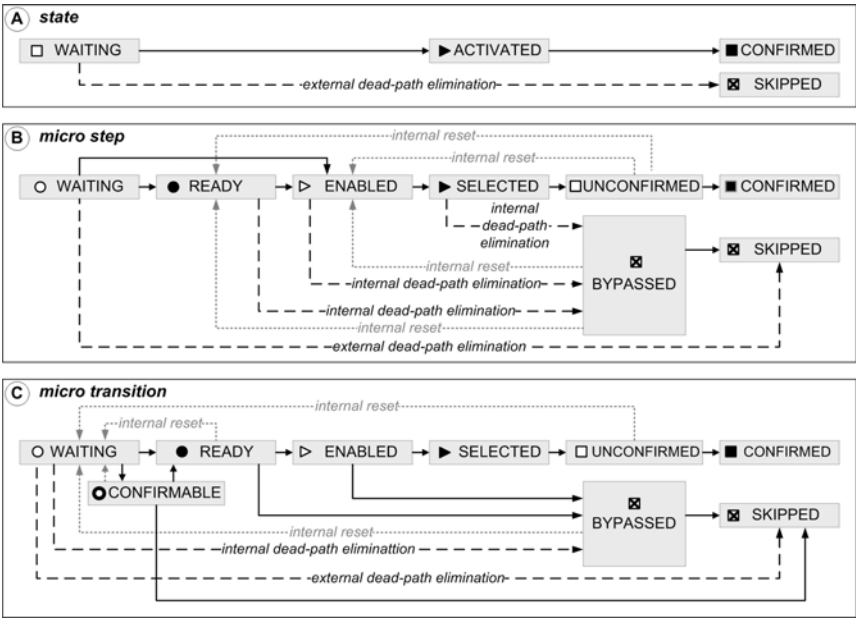


Fig. 4. Operational semantics for states, micro steps and micro transitions

are marked as READY, whereas micro steps not belonging to the start state are marked as WAITING. This differentiation enables us to highlight input fields being relevant for process execution in the currently activated state.

Execution. Starting in state `pending`, micro step `invite` is automatically reached if a value is assigned to the corresponding attribute. Then micro step `invite` is marked as UNCONFIRMED (cf. Fig. 5a). Following this, an employee must provide a value for at least one of the attributes `appraisal` or `appointment`. If for one of these attributes (e.g., `appointment`) a value is set the corresponding micro step is marked as SELECTED (cf. Fig. 5b). The respective micro transition, in turn, is marked as ENABLED. Since no value for attribute `appraisal` is provided (i.e., only one outgoing micro transition is reachable), the priorities of the micro transitions are not relevant. Thus, the ENABLED micro transition can be marked as SELECTED (cf. Fig. 5c). In this case, we omit the other path by performing an *internal dead-path elimination* (cf. Fig. 5d). For this purpose, all micro transitions and steps belonging to the non-selected execution path are marked as BYPASSED (i.e., a micro step is marked as BYPASSED if all incoming micro transitions are marked as BYPASSED).

However, as long as this state change has not been confirmed, an employee may still change attribute settings; i.e., he may want to set the value of attribute `appraisal`. To accomplish this, an *internal reset* of the currently activated state is performed (cf. Fig. 5e). Generally, micro steps and transitions will be reset if an attribute value corresponding to a micro step marked as UNCONFIRMED or BYPASSED is changed. If a value for both attribute `appraisal` and attribute `appointment` is assigned (cf. Fig. 5f) (i.e., more than one micro transition becomes

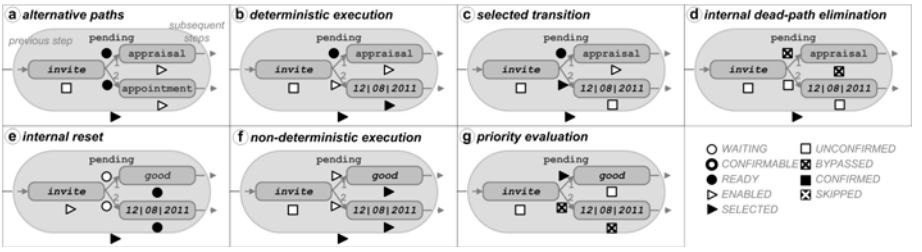


Fig. 5. Execution

ENABLED), we ensure that only one of the micro transitions is actually fired; i.e., always one micro step (and one micro state) can be reached. For this purpose, only the micro transition with the highest priority is SELECTED (cf. Fig. 5g). The other one is marked as BYPASSED using the internal dead-path elimination. If a state is marked as CONFIRMED afterwards, micro steps with marking BYPASSED and transitions are finally marked as SKIPPED.

Changing the state. After a micro step is marked as UNCONFIRMED, outgoing micro transitions are either marked as READY or CONFIRMABLE. More precisely, external micro transitions, for which an explicit user commitment is required, are marked as CONFIRMABLE. Consequently, a mandatory activity enabling this commitment is automatically assigned to the worklist of the responsible user. In our example, after initializing a review, two external, explicit micro transitions are marked as CONFIRMABLE requiring a respective user decision. If one of them is selected, its marking changes from CONFIRMABLE to READY. Opposed to this, implicit micro transitions (internal and external ones) are immediately marked as READY. If an external micro transition is marked as READY, the currently activated state is marked as CONFIRMED. In addition, all corresponding micro steps as well as internal micro transitions (currently marked as UNCONFIRMED) are re-marked as CONFIRMED as well. Following this, the subsequent state (i.e., state pending in our example) is marked as ACTIVATED and its micro steps as READY. The target micro step of the SELECTED external micro transition (i.e., micro step `proposal`) is marked as ENABLED. For this micro step a value of its attribute has to be set Moreover, we perform an *external dead-path elimination* in order to mark micro steps, micro transitions, and states as SKIPPED that can no longer be activated.

Despite any predefined form logic (e.g., sequence) of micro steps, users are allowed to freely choose their preferred execution order; i.e., the order in which required values are assigned to object attributes does not necessarily have to coincide to the one of the corresponding micro steps. In particular, at run-time a micro step can be completed as soon as a value is assigned to its object attribute; e.g., an employee may set the value of attribute `appraisal` although he is guided to first fill in the input field relating to attribute `proposal`. If the value of object attribute `proposal` is set afterwards, the subsequent micro step relating to object attribute `appraisal` is automatically completed. In principle, an entire mandatory activity can be skipped if all required attribute values are assigned in a previous state.

Termination. Finally, execution of a micro process instance terminates if one state containing an end micro step is marked as SELECTED. Opposed to other micro steps, which are marked as UNCONFIRMED, while the state they belong to is marked as ACTIVATED, end micro steps are immediately marked as CONFIRMED. Using the introduced internal and external dead-path elimination, we can ensure that all other states, micro steps and micro transitions are then either marked as CONFIRMED or SKIPPED.

8 Related Work

In [16] we have shown why existing imperative, declarative, and data-driven (i.e., Case Handling [3]) process support paradigms are unable to adequately support object-aware processes. However, to enable consistency between process and object states, extensions of these approaches based on object life cycles (OLC) have been proposed. These extensions include object life cycle compliance [9], object-centric process models [10,11], business artifacts [8], data-driven process coordination [6], and object-process methodology [18]. To be more precise, an OLC defines the states of an object and the transitions between them. Activities, in turn, are associated with pre-/post-conditions in relation to objects states; i.e., the execution of an activity depends on the current state of an object and triggers the activation of a subsequent state. However, none of these approaches explicitly maps states to attribute values. Consequently, if certain pre-conditions cannot be met during run-time, it is not possible to dynamically react to this. In addition, generic form logic is not provided in a flexible way; i.e., there is no automatic generation of forms taking the individual attribute permissions of a user as well as the progress of the corresponding process into account. Finally, opposed to these approaches, PHILharmonicFlows captures the internal logic of an activity as well.

9 Summary and Outlook

In this paper, we introduced the modeling and execution of micro processes which are a fundamental pillar of our PHILharmonicFlows framework for object-aware process management. To enable high flexibility, form-based activities are automatically generated taking the respective user and the current process state into account. Our approach is based on precise rules enabling syntactical correctness as well as on a well-defined operational semantics. Moreover, PHILharmonicFlows goes far beyond the concepts presented in this paper. In future papers we will discuss how to support backward jumps, time events, black-box activities, and specific attribute values. Regarding the latter, for instance, whether or not a particular attribute value becomes mandatory on-the-fly may depend on the concrete value of an attribute belonging to a previous micro step (e.g., an appointment needs only be defined if the employee proposes to invite the applicant). Moreover, future papers will report on the other components of our framework. As example consider *macro processes* which refer to multiple object instances of various object types. Here, issues related to the object-centred coordination and synchronization of related process instances need to be addressed.

References

1. van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M.: Business Process Management: A Survey. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 1–12. Springer, Heidelberg (2003)
2. Silver, B.: Case Management: Addressing unique BPM Requirements. BPMS Watch, 1–12 (2009)
3. van der Aalst, W., Weske, M., Grünbauer, D.: Case Handling: A new Paradigm for Business Process Support. DKE 53(2), 129–162 (2005)
4. Sadiq, S.W., Orlowska, M.E., Sadiq, W., Schulz, K.: When workflows will not deliver: The case of contradicting work practice. In: Proc. BIS 2005 (2005)
5. Künzle, V., Reichert, M.: PHILharmonicFlows: Towards a Framework for Object-aware Process Management. Journal of Software Maintenance and Evolution: Research and Practice (2011)
6. Müller, D., Reichert, M., Herbst, J.: Data-Driven Modeling and Coordination of Large Process Structures. In: Chung, S. (ed.) OTM 2007, Part I. LNCS, vol. 4803, pp. 131–149. Springer, Heidelberg (2007)
7. Gerede, C.E., Su, J.: Specification and Verification of Artifact Behaviors in Business Process Models. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 181–192. Springer, Heidelberg (2007)
8. Bhattacharya, K., Hull, R., Su, J.: A Data-Centric Design Methodology for Business Processes. In: Handbook of Research on Business Process Management, pp. 503–531. IGI Global (2009)
9. Küster, J.M., Ryndina, K., Gall, H.C.: Generation of Business Process Models for Object Life Cycle Compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 165–181. Springer, Heidelberg (2007)
10. Redding, G., Dumas, M., ter Hofstede, A.H.M., Iordachescu, A.: Transforming Object-Oriented Models to Process-Oriented Models. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) BPM Workshops 2007. LNCS, vol. 4928, pp. 132–143. Springer, Heidelberg (2008)
11. Redding, G.M., Dumas, M., ter Hofstede, A.H.M., Iordachescu, A.: A flexible, object-centric approach for business process modelling. In: SOCA, pp. 1–11 (2009)
12. Reijers, H.A., Liman, S., van der Aalst, W.M.P.: Product-Based Workflow Design. Management Information Systems 20(1), 229–262 (2003)
13. Vanderfeesten, I.T.P., 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)
14. Künzle, V., Reichert, M.: Towards Object-Aware Process Management Systems: Issues, Challenges, Benefits. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) Enterprise, Business-Process and Information Systems Modeling. Lecture Notes in Business Information Processing, vol. 29, pp. 197–210. Springer, Heidelberg (2009)
15. Künzle, V., Reichert, M.: Integrating Users in Object-aware Process Management Systems: Issues and Challenges. In: Proc. BPM 2009 Workshop, pp. 29–41 (2009)
16. Künzle, V., Weber, B., Reichert, M.: Object-aware Business Processes: Fundamental Requirements and their Support in Existing Approaches. International Journal of Information System Modeling and Design 2(2) (2010)
17. Dijkstra, E.: A Discipline of Programming. Prentice-Hall, Englewood Cliffs (1976)
18. Dori, D.: Object-Process Methodology. Springer, Heidelberg (2002)

Towards a Method for Realizing Sustained Competitive Advantage through Business Entity Analysis

Matteo Della Bordella[1], Rong Liu[2], Aurelio Ravarini[1],
Frederick Y. Wu[2], and Anil Nigam[2]

[1] Università Carlo Cattaneo – LIUC, Cetic, C.so Matteotti 22, 21053 Castellanza (VA), Italy
[2] IBM T.J. Watson Research Center, 19 Skyline Drive, Hawthorne, New York 10532, USA
{mdellabordella,aravarini}@liuc.it,
{rliu,fywu,anigam}@us.ibm.com

Abstract. Enterprises that succeed in today's highly dynamic business environment often enjoy Sustained Competitive Advantage (SCA) as defined by Barney. While recent strategy literature focuses on exploring various sources of SCA, in this paper, we present an operational method for realizing SCA through Business Entity analysis. Business entity-centric modeling has been a successful approach in rethinking and revolutionizing business operations, in a number of engagements. Our method provides a path from SCA-generating strategies to Business Operations and Business Entities. The resulting Business Operations can be prototyped and analyzed to validate SCA properties. Our approach leverages key constructs from OMG's Business Motivation Model (BMM) and emphasizes the analysis of Influencers – factors that have the capability to impact an enterprise's strategies that generate SCA. Further, these strategies are used to formulate Business Operations that can be defined by Business Entities. IT applications can be generated from the Business Entities using Model-Driven Architecture. Therefore, these discovered Business Entities actually provide a valid scope for innovating Business Operations and developing IT applications that result in SCA for the business.

Keywords: Sustained Competitive Advantage, Model-Driven Architecture, Strategy, Goal, Objective, Operation, Influencers, Business Entity.

1 Introduction

Nowadays, more than in the past, enterprises are facing new and diverse challenges posed by turbulent environments, and are required to deliver complex mixes of products and services that are rapidly adaptable to the needs of changing markets [3]. The ability of a business to adapt rapidly and efficiently in response to changes is referred to as business agility [27]. However, business agility is not about reacting passively when the case for change becomes desperately obvious in an enterprise, but rather innovating strategically its core business to achieve *sustained competitive advantage* (SCA) [2]. A firm is said to have SCA when its competitors cannot imitate its core strategy or are unable to duplicate the benefits of the strategy. Therefore, it is critical

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 216–230, 2011.

to develop the right business strategies, to revolutionize business operations accordingly, and to implement appropriate IT systems to support the operations.

While identifying which strategies can generate SCA is one issue, another issue is to design business operations and IT systems which can be well aligned with the strategies. Model-Driven Architecture (MDA) [21] is a top-down approach to modeling business and creating cost effective IT solutions through model transformation. MDA provides three levels of modeling abstractions: (1) a computation-independent model (CIM) oriented to model business independent of underlying software models or implementations; (2) a platform-independent model (PIM); and (3) a platform-specific model (PSM). Architectural mappings between layers are specified to enable code generation to accelerate application development. To close the business-IT gap, typically, a company's strategic vision needs to be captured precisely in a CIM and transformed into a PSM and IT applications in a top-down approach. However, it is often challenging to capture a strategic vision and faithfully transform it into IT applications.

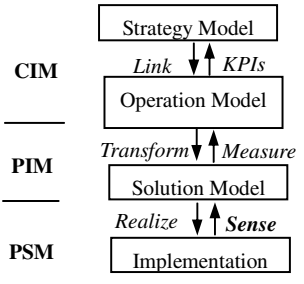


Fig. 1. The Model Driven Architecture

Model-Driven Business Transformation (MDBT) [16, 17] extends MDA by dividing CIM into two sub-layers, Strategy Model and Operation Model, as shown in Fig. 1. To ensure a successful top-down transformation, the Strategy Model captures business goals and the Operation Model describes business operations for achieving the goals. For example, Balanced Scorecard [15] is a well-known method for strategy modeling. The business entity-centric approach is an effective technique for modeling Business Operations [5, 19]. Unlike traditional business process modeling, this approach unifies the modeling of business processes and information, thereby creating a holistic model of operating the business. A *business entity* (a.k.a. business artifact), manifesting an operational goal, is characterized by a self-contained information model and a streamlined lifecycle model. The lifecycle model consists of a collection of business activities that act on the business entity progressing towards an operational goal. The information model includes information required to execute the activities as well as the results produced. For example in the account opening operation in a bank, the data entity *Arrangement* is likely to be identified as a business entity, as shown in Fig. 2. Its lifecycle model describes business activities such as *Identifying Customers*, *Proposing Arrangement*, *Accepting Arrangement*, and *Activating Arrangement* etc. Each of these activities accomplishes a significant milestone in the lifecycle of *Arrangement*. The information model of this business entity consists of

data attributes, some of which might be references to other business entities e.g. *Arrangement* has *Requirement* and *Customer* (which is a reference to another entity that represents the customer). Clearly there are similarities between business entities and data objects in Object-oriented analysis and design (OOAD) [7]. However, different from OOAD which focuses on fine-grained data objects, the business entity-centric approach captures the deep structure of data objects, groups them into a few business entities to manifest business operational goals [18]. A business entity-centric operation model can be automatically transformed into a solution model and then an executable implementation model [5,16].

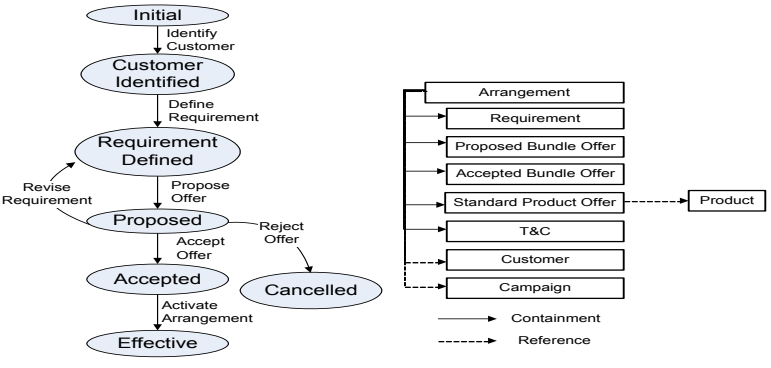


Fig. 2. Business Entity's lifecycle and Information Model

In general, a company needs to set a number of goals, from different perspectives, as stated in a Balanced Scorecard. However, constrained by limited resources, an enterprise always needs to prioritize its goals to select those that can contribute to SCA. Unfortunately, MDA does not provide an approach to prioritizing business goals but rather assumes that business goals have been decided and well described. Other approaches [2, 15, 24, 26] often emphasize business strategies, but give little consideration to the linkages to business operations and downward transformation to IT applications. MDBT also lacks a formal approach to identifying the business goals, operations, and business entities that together contribute to SCA.

In this paper, we bridge the gap by providing a methodology for modeling business goals and determining whether a goal can generate SCA. Furthermore, following the MDA approach, we identify operations and business entities from such a goal to provide a valid scope for innovating business operations and developing IT applications that result in good alignment between business and IT. This method leverages constructs that have been standardized by the Object Management Group in its Business Motivation Model (BMM) [20]. A key advantage of utilizing this connection with BMM is that vendor applications are motivated to conform to BMM, and can as such be useful for realizing the results of the method developed here. We carefully analyze two case studies in the literature to illustrate our methodology.

After operations and business entities are identified, the MDBT approach can be followed to elicit information related to these concepts and eventually realize these concepts through the generation of requirements for processes and the IT applications

needed to support the processes [5]. Thus, the creation of IT applications from the identified operations and business entities is out of the scope of this paper. Also, this paper focuses on theoretical development of our methodology. The evaluation of this methodology through real engagements has been planned but is not covered here.

The remainder of the paper is organized as follows. In section 2 we describe the theoretical background of our research, including the Business Motivation Model, the concepts of Sustained Competitive Advantage, and the Resource Based View of the firm theory. In section 3 we describe the proposed methodology with an illustrating example. We further demonstrate this methodology using a case study in Section 4. Section 5 compares our approach with the related work. Finally, Section 6 discusses study limitations and concludes with a brief description of our future research.

2 Theoretical Background

2.1 Business Motivation Model (BMM)

The Business Motivation Model [20] is a standard adopted by the Object Management Group (OMG) in 2005, and the current version (1.1) was published in May, 2010. The BMM is a structure for developing, communicating, and managing business plans. We chose to build on the BMM because its structure encompasses top level strategic concepts such as Goals, Objectives, and Resources, but its scope ends at the boundary with Business Operations. Furthermore, the BMM has a formal metamodel that includes a vocabulary catalog of concept definitions, and has been adopted by a major standards organization. However for the purpose of our work we retained necessary to integrate BMM with some basic concepts taken from RBV that will be explained later and we are aware that some of the definitions contained in the BMM document, such as the one of Strategy or Influencer, are not universal and can provide a starting point for a discussion.

Fig. 3 shows the elements of the BMM that are most relevant to our discussion. At the top is an End, which is very generally something that is to be accomplished. One specialization of End is Desired Result, which is a target that an enterprise intends to maintain or sustain. There are two types of Desired Result: *Goal* and *Objective*. A Goal is a state to be brought about by appropriate means. An Objective is a specific time-targeted, measurable, attainable target that an enterprise seeks to meet. An Objective quantifies a Goal.

Also at the top of the BMM diagram is Means. A Means is anything that may be called upon, activated, or enforced to achieve Ends. For our purposes, the most relevant type of Means is a Course of Action, defined as a plan for configuring some aspect of an enterprise undertaken to achieve Ends. Of the types of Courses of Action, the most significant here is *Strategy*, which is an element of a plan devised through the science and art of business leadership exercised to ensure the most advantageous conditions [20].

The next relevant element in the BMM is *Influencer*. BMM defines an Influencer as something that is neutral but has the capability to impact the enterprise in its employment of Means or achievement of its Ends [20]. Specifically, an Influencer can affect the Strategy of an enterprise. Influencers can be internal to the enterprise or

external if they are outside the enterprise's organizational boundaries. For example, Competitor, Customer, Regulation, Partner, Environment and Technology are categories of *External Influencers*. *Internal influencers* are those that are available internally (under the control of the enterprise) for carrying out the business of the enterprise. Corporate Value, Infrastructure, Assumption, Habit, Issue, Management Prerogative and Resource are categories of Internal Influencer. Note that, in the BMM specification document [20] there is an important statement about Influencer that we will utilize, but which is not actually shown as a formal relationship in the diagrams of the BMM specification. The statement "An Influencer is something that can cause changes that affect the enterprise in its employment of its Means or achievement of its Ends" implies a direct relationship between Influencer and the Means of achieving Ends. This implies the relationship shown in Fig. 3, that an Influencer can affect a Course of Action. To summarize the relevant relationships we will use from the BMM, *an Influencer can affect the Strategy that an enterprise uses to channel its efforts toward the achievement of its Goals and Objectives*.

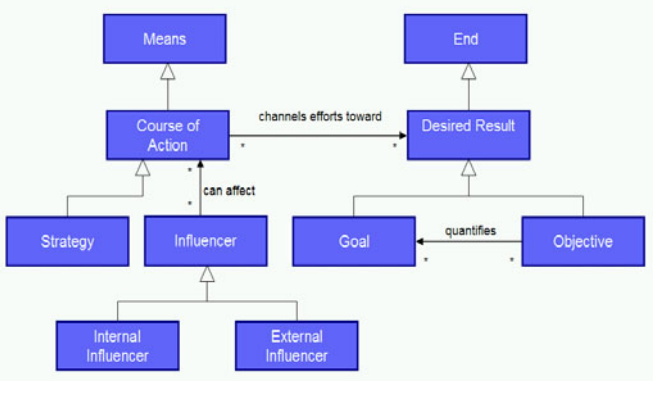


Fig. 3. UML Class Diagram of Business Motivation Model (BMM) (Adapted)

2.2 Sustained Competitive Advantage (SCA) and Resource Based View

The Resource Based View of the firm theory (RBV or RBT) [2] proposed by Barney in 1991 has the objective to understand how a company can achieve a *Sustained Competitive Advantage* (SCA) by implementing strategies that exploit internal strengths, through responding to environmental opportunities, while neutralizing internal threats and avoiding internal weaknesses. SCA can be achieved through *firm resources*. This view brings out two important concepts, SCA and firm resources. *A firm is said to have a sustained competitive advantage when it is implementing a value creating strategy not simultaneously being implemented by any current or potential competitors and when these other firms are unable to duplicate the benefits of this strategy* [2]. In this view, firm resources are defined as "all assets, capabilities, organizational processes, firm attributes, information, knowledge, etc. controlled by a firm that enable the firm to conceive of or implement strategies that improve its efficiency and effectiveness", and a firm resource that has the potential to provide SCA must have the following four characteristics:

- *Valuable* – enabling a firm to conceive of or implement strategies that improve its efficiency and effectiveness,
- *Rare* – not possessed by a large number of competitors or potential competitors,
- *Imperfectly imitable* – cannot be obtained by firms that do not possess it, and
- *Not substitutable* – there are no strategically equivalent resources that are themselves not rare or imitable.

On the definition of SCA, apart from some discussions about the sustainability and duration of the competitive advantage [29] there is a good level of agreement among scholars. Motivated by RBV, in this paper, we propose a new methodology to investigate the sources of SCA. Following the definition of SCA which really emphasizes strategies, in this paper, we examine two specific aspects of strategies for the source of SCA: (1) specific internal or external factors that have the capability to impact strategies of an enterprise, and (2) business behaviors that realize strategies. The first aspect corresponds to Influencers in BMM. Inspired by the analysis of firm resources, we also examine if an Influencer has the same characteristics to determine its potential to generate SCA through strategies. It may be arguable whether Influencer is a concept equivalent to firm resource as defined RBV, but this issue is beyond the focus of this paper. The second aspect is referred to as business operations. Since the business entity approach has been used effectively for modeling business operations [5,8], we indirectly create the linkage between SCA and business entities. Accordingly, an enterprise can use business entities as a vehicle for revolutionizing operations and developing supporting IT applications. Therefore, different from conceptual analysis of firm resources as the source of SCA, our proposal aims to provide an actionable method to help enterprises choose the strategies that generate SCA.

3 Methodology: Sustained Competitive Advantage Using Business Entities (SCUBE)

In this section, we describe our methodology for choosing strategies that generate SCA (i.e. *SCA-generating Strategies*) through analyzing Influencers and then business operations and Business Entities that realize these strategies. We illustrate the methodology, called "*Sustained Competitive Advantage Using Business Entities*" (SCUBE) through a case study from the newspaper industry. It should be noted that we do not suggest that automation of this method is feasible, as most steps require deep business knowledge. Fig. 4 shows that SCUBE has six steps listed as follows.

Methodology: SCA Using Business Entities (SCUBE)

Input: Defined Desired Results, Strategies that channel efforts towards the achievement of the Desired Result, and Influencers using the adapted BMM model (see Fig. 3).

Output: Business Operations responsible for SCA and Business Entities produced by these Operations

Step 1: Examine if an Influencer or a combination of Influencers is valuable, rare, imperfectly imitable, and not substitutable. Select the Influencers that satisfy these properties.

Step 2: Describe Strategies that are impacted by the Influencers selected in Step 1. These are SCA-generating strategies.
Step 3: Identify Goals that are achieved by these SCA-generating Strategies.
Step 4: Identify Objectives which quantify each of the Goals.
Step 5: Identify Operations that achieve these Objectives
Step 6: Identify Business Entities produced by these Operations. Thus, these Business Entities can lead to SCA.

The **input** to our proposed methodology is Desired Results (Goals and Objectives), Strategies, and Influencers specified following the adapted BMM model shown in Fig. 3. Traditional methods, for example, Balanced Scorecard [15] can be used to help an enterprise define Goals and Strategies to achieve these Desired Results. Note that, often, it may be difficult to define Strategies clearly without a thorough assessment of Influencers. Influencers have the capacity to shape strategies. By assessing Influencers with respect to their impact on Ends or Means, an enterprise can understand its strengths, weaknesses, opportunities and threats [20] and then be able to refine its Strategies. The **output** of SCUBE is the identified Business Operations mainly responsible for SCA and Business Entities produced by them.

It is worth noting that the key to our methodology is to identify SCA-generating Strategies. By its definition, SCA depends on Strategies and on their benefits. According to BMM, Influencers have a direct impact on Strategies. Thus, motivated by RBV theory, we propose that *if a Strategy is shaped by Influencers that possess the same four characteristics as firm resources in RBV, viz. value, rarity, imperfect imitability and non-substitutability, then this Strategy is able to generate SCA*. Now we need to investigate which category of Influencers potentially possesses these characteristics. While Internal Influencers can possibly possess all of these four characteristics *per se,* External Influencers (e.g. environment, technology, and regulations), by their nature, cannot be rare as they are outside the boundaries of an enterprise and can be possessed by other competing companies. Thus a SCA-generating Strategy cannot be exclusively caused by an External Influencer. However, it may result from a particular attitude or behaviour that a firm has towards that Influencer. Therefore, a SCA generating Strategy can be derived from the combination of an Internal Influencer such as, Habit, Assumption, Corporate Value, and Management Prerogative, with an External influencer such as Environment, Technology, and Regulations. An example will be provided to illustrate this point. Moreover there are some kinds of Influencers, such as Issue, that can negatively affect a Strategy and which therefore are not able to shape a SCA-generating strategy.

In order to give a better explanation of the methodology, we provide an example taken from the publishing industry, about "El Norte", a pioneering firm in the information industry in Mexico (for further information on the case see [14]). Headquartered in Monterrey, El Norte in 1997 had three main outlets: *El Norte News,* the number one newspaper in Monterrey, *Reforma*, the number two newspaper in Mexico City and the only global Mexican newspaper, and *Infosel*, a Monterrey-based provider of on-line, real-time information for businesses. The managers in El Norte considered the vision of this company to be "an information company with different channels to distribute the information." Table 1 provides a list of the main internal Influencers

identified. Table 2 summarizes our analysis results of El Norte example following the methodology. Next, we describe the result of each step in detail.

Following the methodology, in **Step 1**, we examine the Influencers identified in Table 1. We find that "*connectedness with Mexican Bolsa*", as Management Prerogative, is a valuable, rare, not substitutable and imperfectly imitable internal influencer and can have direct impact on a SCA generating strategy considered *per se.*

Thus, **Step 2** describes Strategies shaped by the Influencers selected in Step 1 and considers them as SCA-generating Strategies. So in our example, the Influencer "Connectedness with Mexican Bolsa" would be exploited to implement a SCA-generating Strategy that is "*cooperate in real time with Mexican Bolsa and Government to get all the latest information.*"

Step 3 is the identification of Goals produced by the SCA-generating Strategies. The purpose of this step is to link Strategies with Goals. As shown in Fig. 3, because Strategy is a specialization of Course of Action and Goal is a specialization of Desired Result, the association between Course of Action and Desired Result can be inherited by Strategy and Goal. Thus, Strategies channel efforts towards achieving Goals. Furthermore, we take Sustained Competitive Advantage as a Boolean attribute of Desired Result. Thus, Desired Results which yield SCA are those produced by the SCA-generating Strategies described in Step 2. For example, from the previously stated Strategy "cooperate real time with Mexican Bolsa and Government to get all the latest information", we believe these two Goals deliver SCA: "*to be the leader in distributing information among Mexican finance experts*" and "*offer a daily issue with the latest information available about Mexican finance*." These are consistent with the vision of the company: "to be the leader in distributing information in Mexico."

Subsequently **Step 4** is the identification of Objectives which quantify the Goals defined in Step 3. According to BMM, An Objective is measurable and can be attained with a time frame. In our example such an Objective would be "*By 1st January 1999, offer a special issue every Monday about Mexican Bolsa*". Two other objectives are also identified, as shown in Table 2.

In **Step 5,** we identify Operations needed to achieve each Objective. Business Operation is a holistic description of what the business needs to do (process, information and roles and systems) to achieve an objective. A systematic approach would consider each objective, and make subjective assessment of how well each process helps to achieve the objective. In the current example, the Objective *("By 1st January 1999 … Bolsa")* from Step 4, can be achieved through the *newspaper publishing* Operation. Usually Objectives will state desired performance of what the business needs to do, thereby providing clues about the Operation (see example in next paragraph). Table 2 shows the other two Operations identified.

Finally, **Step 6** identifies Business Entities produced by the Operations identified in Step 5, and subsequently builds models of these Business Entities, including lifecycle and information models. Once we have clearly defined Operations that allow a company to meet the Objectives, the identification of related Business Entities becomes straightforward as by definition each Operation involves at least one subject that is a Business Entity. A Business Entity encapsulates knowledge concerning the progress toward achieving the goal of Operations [5, 19]. Since each Objective quantifies a Goal, the Business Entities must collectively supply the data needed to compute Objectives. For example, the newspaper publishing Operation in the case study

produces *Monday Issue*, which therefore is identified as a Business Entity. Metrics related to *Monday Issue* contribute to calculation of the corresponding Objective (just a Boolean in this case). Newspaper Package and Marketing Campaign are Business Entities identified through the other Operations, as shown in Table 2.

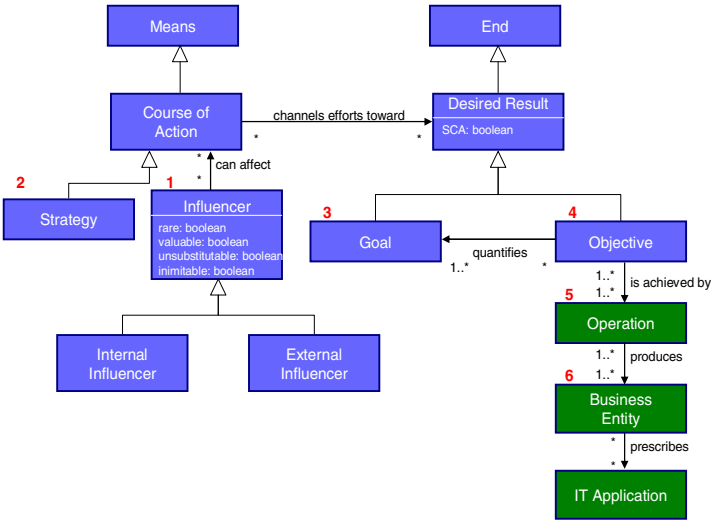


Fig. 4. Modified BMM Class Diagram with SCUBE Methodology

Figure 4 shows the main steps of the SCUBE methodology. This figure extends from BMM model (see Fig. 3) by adding additional elements, SCA, Operation, and Business Entity, as highlighted in green.

Table 1. Internal Influencers in "El Norte" example

Influencer	Example
Habit	El Norte's employees are typically young, health conscious and hard-working; Infosel provided free seminars to educate its clients
Infrastructure	The headquarters is in Monterrey; a dedicated distribution channel; a private IT infrastructure
Issue	Maintain Infosel's independency from US companies; How to educate clients to read news on internet
Management Prerogative	Openness to IT technologies and innovation; connectedness with Mexican Bolsa and government
Resource	CEO Alejandro Junco; young and motivated personnel; strong community network

Note that by applying Steps 3 to 6 to all identified SCA-generating Strategies, the output of SCUBE is inevitably a partial picture of the business, as SCUBE is designed to identify just those Business Entities influencing the creation of SCA, but does not intend to discover all Business Entities involved in the business of a company.

The steps following the proposed methodology are designed to follow the MDBT framework for the operational modeling of the identified Business Entities and then realization of the operational model through IT Applications. Moreover, an enterprise needs to evaluate IT dependent strategic initiatives involving the Business Entities, and in particular, exploit and build a solid roadmap to the SCA. The rapid prototyping of the resulting Operations provides the mechanism for validating, or even rethinking, the business approach. The identified Business Entities can themselves be assessed for SCA characteristics, and may suggest modifications that would make the case for SCA even more compelling.

Table 2. Illustrating Example – El Norte

Step	Element	Example		
1	Influencer or combination	Connectedness with Mexican Bolsa and government (valuable, rare, imperfectly imitable, not substitutable)		
2	SCA-generating Strategy	Cooperate in real-time with Mexican Bolsa and Government to get all the latest information		
3	Desired result and goal	To be the leader in distributing information among Mexican finance experts. Offer a daily issue with the latest information available about Mexican finance		
4	Objectives	By 1st January 1999 offer a special issue every Monday about Mexican Bolsa	By 1st January 1999 sell 500000 copies per day of *Infosel*	By 1st January 1999 subscriptions to *Infosel* for a total revenue of 5 billion dollars
5	Operation	Newspaper publishing	Delivery process	Newspaper marketing
6	Business Entity	Monday Issue	Newspaper Package (content)	Marketing Campaign

So far we have described a path from SCA-generating Strategies to Operations and Business Entities and then to IT applications. With model-driven methods, it can be challenging to handle changes that occur at different levels (see Figure 1) [9]. Since our methodology consists of multiple interconnected model elements (see Figure 4), changes can be handled incrementally. We first identify which model elements are affected by a change. Then for each of these model elements, we investigate whether its associated elements are affected. Eventually, the change is propagated to all affected elements to make all model elements consistent.

4 An Example of the Methodology

As a second example of the application of our methodology, we consider the case study written by Michael Hammer [12] on Progressive Insurance, a US automobile insurer. Our analysis is shown in Fig. 5. The focus of Hammer's article is "operational innovation," and the specific innovation created by Progressive was the Immediate Response claims handling process. We can identify this as an Influencer (Step 1) because it clearly affected the way in which Progressive achieved its goals. Specifically this is an Internal Influencer that falls in the category of Corporate Value, because this is a practice or an idea promoted by the company, in accordance with the definition of Corporate Value in the BMM standard [20]. We next test this Influencer

against the four required attributes for SCA. Its value is due to lower costs and improved service quality, and its rarity was guaranteed because it was (at the time) unheard of to have an adjuster inspect a damaged vehicle within a day of claim notification. This capability is very difficult to imitate because it requires major investments (e.g. mobile claims vans) and major cultural changes in the workforce. There is no practical substitute for this unique business process. Hence, we conclude that Immediate Response is an Influencer that can support Sustained Competitive Advantage. This operational innovation was developed with a strategy in mind, and that strategy can be identified (Step 2) as "Streamline Claims Processing."

According to the case study, streamlining claims processing has two Goals (Step 3): lowering costs and premiums, and increasing customer retention. As illustrated in Fig. 5, these can both be seen as sub-Goals of a major Goal "Grow Market Share." Customer Retention can be increased by increasing Customer Satisfaction, which in turn can be increased by shortening the claim processing cycle. We then need to quantify the sub-Goals. Although these were not explicitly stated in the article, it is not difficult to define plausible Objectives (Step 4) consistent with the case study. For example, for customer retention, we define an Objective to have an adjuster visit the claimant within 9 hours in at least 90% of all claims. For lower costs, we could have an Objective of achieving 96% Combined Operating Ratio (claims and operating expenses as a percentage of premium income). The first Objective is a measure of performance of the Claims Processing operation, while the second Objective is a measure of performance of both the Claims Processing and the Underwriting Operations (Step 5). For the final step in our methodology, we identify the key Business Entities associated with these operations as Insurance Claim and Insurance Policy, respectively.

In a similar manner, Fig. 5 shows the analysis of two additional Influencers described in the Hammer article: integration with competitors' websites for pricing information and novel analysis of customer risk profiles. We consider the first of these to fail the inimitability test, because it is straightforward and inexpensive for a competitor to build the same price comparison capability. As for risk profile analysis, it is reasonable that Progressive may have developed undisclosed risk analysis algorithms that would be valuable in reducing risk, and difficult for competitors to duplicate or imitate, at least for some period of time. Consequently we consider this Influencer to be another source of SCA.

Besides Influencers shown in Fig. 5, there are other Influencers in this business. For example, a CRM system can be classified as an Internal Influencer in the category of Resource, but it is not rare, as other competing companies can also have equivalent CRM systems. Hence, although this Influencer has impact on Strategy "offer customized policies to customers" (Step 2), this Strategy does not lead to SCA as competitors are able to apply the same Strategy and receive the same benefits.

Moreover, as stated previously, a combination of Influencers can support SCA as well. Consider, for example, "openness towards innovation", an internal Influencer in the category Management Prerogative. This Influencer does not possess the four requisites for SCA either, because this particular management attitude is theoretically imitable by other competing companies. But if we combine this Influencer with an external Influencer, "insurance companies generally tend to have a rigid claim processing process", in the category Environment, we can conclude that the combination is valuable, rare, imperfectly imitable and not substitutable, because this external

Influencer indicates that in fact the competing companies are not able to imitate the Progressive's "openness to innovation" as their structure is rigid, and the internal Influencer is rare because the other competing companies are not very open to innovation. Certainly, openness towards innovation is also valuable and not substitutable. Therefore, in this particular context this combination of Influencers possesses the 4 required attributes for SCA.

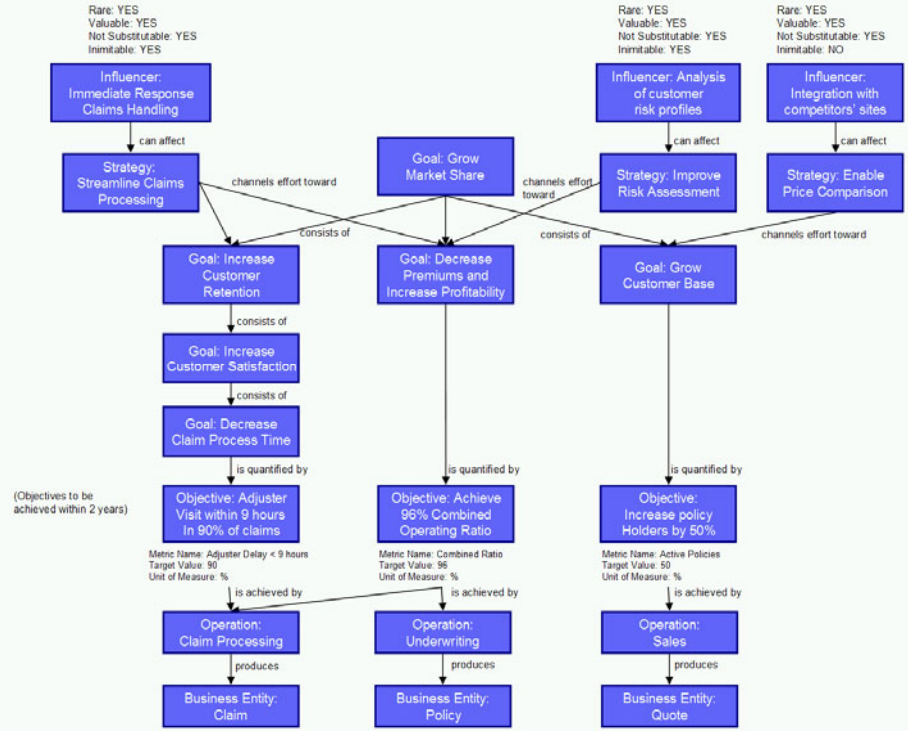


Fig. 5. The Progressive Insurance example

This second example further illustrates the usefulness of our methodology in determining the key Business Entities that are worthy of focused development. This example also demonstrates that in some cases, the key Business Entity is precisely the object on which the Influencer has influence, i.e. Immediate Response (the Influencer) changes the way that Claims (the Business Entity) are processed.

5 Related Work

Understanding sources of Sustained Competitive Advantage has always been a research area of great interest in the Strategy field [2,4,11,14, 29], while business agility [1, 3, 13, 30] is an emerging topic of growing interest among scholars and practitioners. Even though SCUBE methodology represents a new and original combination of these two areas, there are some other notable studies directed towards the achievement of business agility through the strategic analysis of the business.

Component Business Model (CBM) [22, 25] is a strategic analysis methodology with the goal of process optimization. It provides a graphical representation of an enterprise, divided in sub-units called components, and allows identifying components which need to be improved to achieve deep business differentiation. Business Motivation Model [20], as we described before, provides a structure for developing, communicating, and managing business plans in an organized manner. Specifically, it identifies elements inter-related with Strategy. However, it lacks techniques to prioritize Strategies and scope Operations for SCA.

Another thread of related work is requirements engineering. Recently, Goal-Oriented Requirements Engineering (GORE) [28] approaches receive great attention. GORE approaches focus on the elicitation, refinement and analysis of goals of the software rather than on the formulation of software system requirements. Especially i* [31], an agent-oriented modeling framework used for requirements engineering, proposes strategic models that capture the environment of the future software system including the stakeholders of the system, their objectives, and their relationships. These strategic models are used to understand why a new system is needed and what new system configurations are needed to meet user requirements. Another notable approach is B-SCP [6] that integrates strategies, context, and processes to define IT requirements. Our methodology differs from these approaches in that it attempts to align IT requirements with SCA-generating strategies.

Moreover within the business strategy field, there are plenty of strategic analysis methodologies that can be employed within companies such as Balanced Scorecard [15], Lean Sigma [10], Critical Success Factors [26] or Core Competencies [24], but none of them proposes a practical approach that can be employed to achieve business agility. Instead, our methodology, embracing the RBV theory and the concept of Sustained Competitive Advantage, provides an operational approach to aligning strategies, operations, and IT together in achieving SCA and then business agility.

In the area of operational modeling, the business entity-centric approach has been proposed and tested in a number of engagements [5, 8]. However, the current practice identifies Business Entities through intense consulting sessions. Those sessions are time consuming and demand consulting skills. Moreover, there is no systematic approach to ensuring the alignment between Strategy and Operational modeling. Our work bridges exactly this gap.

6 Conclusions and Future Research

In this paper we propose a new methodology for achieving Sustained Competitive Advantage synthesizing and evolving concepts from Resource Based View of the firm, Business Motivation Model and MDBT. We illustrate the methodology through two case studies from the literature. Of course the real test of the methodology will be its use in a real engagement.

We have a body of evidence that the MDBT framework and thinking has resulted in improving the operations of businesses significantly. The SCUBE method described here provides the formal linkage to strategy that has been lacking in MDBT; the expected pay-off of course is business agility. A remarkable aspect of the proposed methodology is that it is very concrete in the sense that we propose a practical way that would allow any company to achieve business agility with a set of formalized steps. None of the other

paradigms or models mentioned within the paper possesses this characteristic, as BMM is in fact just a model of a company's business and SCA is a principle that drove our work. Our method, properly integrated with MDBT can practically help organization in developing business solutions from strategy; development of business solutions from technological innovation is not the focus of our method.

We are aware that the methodology we proposed has some limits and needs to be tested in real applications. An empirical research on the SCUBE methodology with multiple case studies is the next step of our research and our main priority An interesting point is certainly related to the applicability domain of the SCUBE methodology: theoretically nothing precludes an application in large, medium or small companies or in companies operating in different business sectors and contexts, but practically we might discover that SCUBE suits perfectly some environments while has some lacks in others. The methodology should include definition of a performance measurement system. A monitoring system should be put in place to provide executives with up-to-date information about the strategic impact of the initiatives performed on the business entities and the software applications that support them. This final stage of the methodology has not been defined yet. We are currently studying the possibility of adapting some popular business measurement technique such as the Balanced Scorecard.

References

1. Abrahamsson, P., Hanhineva, A., Jäälinoja, J.: Improving Business Agility Through Technical Solutions: A Case Study on Test-Driven Development in Mobile Software Development. In: IFIP International Federation for Information Processing, vol. 180, pp. 227–243 (2005)
2. Barney, J.: Firm resources and sustained competitive advantage. Journal of Management 17(1), 99–120 (1991)
3. Bennet, A., Bennet, D.: Organizational Survival in the New World: The Intelligent Complex Adaptiven System. Elsevier Science, Burlington (2004)
4. Bharadwaj, A.S.: A Resource-Based Perspective on Information Technology Capability and Firm Performance: An Empirical Investigation. MIS Quarterly 24(1), 169–196 (2000)
5. Bhattacharya, K., Caswell, N.S., Kumaran, S., Nigam, A., Wu, F.Y.: Artifact-centered operational modeling: lessons from customer engagements. IBM Systems Journal 46(4), 703–721 (2007)
6. Bleistein, S.J., Cox, K., Verner, J., Phalp, K.T.: B-SCP: A requirements analysis framework for validating strategic alignment of organizational IT based on strategy, context and process. Information and Software Technology 48(9), 846–868 (2005)
7. Booch, G.: Object-oriented Analysis and Design with Applications, 3rd edn. Addison-Wesley, Reading (2007)
8. Chao, T., Cohn, D., Flatgard, A., Hahn, S., Linehan, M., Nandi, P., Nigam, A., Pinel, F., Vergo, J., Wu, F.y.: Artifact-based transformation of IBM global financing. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 261–277. Springer, Heidelberg (2009)
9. France, R., Rumple, B.: Model-driven Development of Complex Software: A Research Roadmap. In: Proceeding of 2007 Future of Software Engineering, FOSE 2007 (2007)
10. George, M.L.: Lean Six Sigma - combining Six Sigma quality with Lean speed. McGraw-Hill, New York (2002)
11. Grant, R.M.: The resource-based theory of competitive advantage: implications for strategy formulation. California Management Review 33(3), 114–135 (1991)

12. Hammer, M.: Deep Change: How Operational Innovation can transform your Company. Havard Business Review, 84–93 (April 2004)
13. Highsmith, J., Cockburn, A.: Agile software development: the business of innovation. Computer 34(9), 120–127 (2001)
14. Jarvenpaa, S.L., Leidner, D. E.: An information company in Mexico: extending the resource-based view of the firm. In Proceedings of the Eighteenth International Conference on Information Systems, ICIS (1997)
15. Kaplan, R., Norton, D.: The Balanced Scorecard – Measures That drive Performance. Harvard Business Review, 71–80 (1992)
16. Kumaran, S.: Model Driven Enterprise. In: Proceedings of Global Integration Summit 2004, Banff, Canada (2004)
17. Lee, J.: Model-Driven Business Transformation and Semantic Web. Communication of ACM 48(12), 75–77
18. Liu, R., Wu, F.Y., Kumaran, S.: Transforming Activity-Centric Business Process Models into Information-Centric Models for SOA Solutions. Journal of Database Management 21(4), 14–34 (2010)
19. Nigam, A., Caswell, N.S.: Business Artifacts: An approach Operational Specification. IBM Systems Journal 42(3), 428–445 (2003)
20. OMG Business Motivation Model (BMM) Version 1.1, http://www.omg.org/spec/BMM/1.1/ (2010-05-01)
21. OMG Model Driven Architecture (MDA) Guide Version 1.0.1, http://www.omg.org/cgi-bin/doc?omg/03-06-01 (March 6, 2001)
22. Pohle, G., Korsten, P., Ramamurthy, S.: The specialized enterprise: A fundamental redesign of firms and industries. IBM Institute for Business Value (March 2005), http://www-1.ibm.com/services/us/index.wss/ibvstudy/imc/a1009224?cntxt=a1005266
23. Prahalad, C.K., Hamel, G.: The Core Competence of the corporation. Harvard Business Review, 79–91 (May 1990)
24. Porter, M.E.: What Is Strategy. Harvard Business Review 32(1), 10–20 (1996)
25. Ramamurthy, S., Robinson, M.S.: Simplify to Succeed: Optimise the customer franchise and achieve operational scale: Retail financial institutions in 2005. IBM Business Consulting Services (2005), http://www-8.ibm.com/services/pdf/gw510-9108-00_fs_exec.pdf
26. Rockart, J.F.: The changing role of the information systems executive: a critical success factors perspective. Sloan Management Review (Fall 1982)
27. Tsourveloudi, N., Valavanis, K.: On the Measurement of Enterprise Agility. Journal of Intelligent and Robotic Systems 33, 329–342 (2002)
28. Van Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Roundtrip from Research to Practice. In: Proc. International Conference on Requirements Engineering (RE 2004), Kyoto, Japan (September 2004)
29. Wade, M., Hulland, J.: Review: The Resource-Based View and Information Systems Research: Review, Extension, and Suggestions for Future Research. MIS Quarterly 23(1), 107–142 (2004)
30. Weill, S., Broadbent, M.: IT Infrastructure for Strategic Agility, Working papers from Massachusetts Institute of Technology (MIT), Sloan School of Management (2003), http://dspace.mit.edu/handle/1721.1/1831
31. Yu, E., Liu, L.: Modelling trust for system design using the *i** strategic actors framework. In: Falcone, R., Singh, M., Tan, Y.-H. (eds.) AA-WS 2000. LNCS (LNAI), vol. 2246, p. 175. Springer, Heidelberg (2001)

Business Process Configuration Wizard and Consistency Checker for BPMN 2.0

Andreas Rogge-Solti, Matthias Kunze, Ahmed Awad, and Mathias Weske

Hasso Plattner Institute at the University of Potsdam
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany
{andreas.rogge-solti, matthias.kunze, ahmed.awad,
matthias.weske}@hpi.uni-potsdam.de

Abstract. A rapidly changing environment, in terms of technology and market, forces companies to keep their business processes aligned with current and upcoming requirements. This is still a major issue in modern process oriented information systems, where improvements on process models require considerable effort to implement them in a technical infrastructure.

We address this problem by lifting technical details into BPMN 2.0 process models and present a configuration wizard for these process models in the open-source modeling tool Oryx. This wizard includes a consistency checking mechanism to automatically discover inconsistencies in the data dependencies of a process model. Immediate feedback after changes to the model eliminates a crucial source of errors when configuring or redesigning business process models, leading to more efficient process implementation.

Keywords: consistency checking, business process redesign, business process configuration, tool support.

1 Introduction

Within the last decade, business process management gained increasing impact on the way organizations conduct their businesses. Business process management is the key instrument to understanding and organizing the activities an organization undertakes to produce goods or deliver services. As businesses and their environment undergo continuous dynamics, the processes of an organization have to be steadily improved. However, adopting changes is still one of the biggest issues of process oriented information systems [12] and process management tools provide only limited support.

In this paper we present a configuration wizard that uses a model driven approach to leverage graphical process models, enrich them with technical details, and automatically generate process representations that can be readily enacted by a process engine. The wizard is built into the open-source modeling tool Oryx and includes a consistency checker for data flow in the process. The focus of this approach is on graphical models, since changes are most easily and consistently performed on a graphical representation. This enables an easy configuration of process models and quick redesign of existing process models.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 231–245, 2011.

The remainder of the paper is organized as follows. Section 2 introduces preliminary definitions. Section 3 elaborates on particular problems of the traditional approach to business process configuration. Section 4 presents conceptual ideas to solve these issues. In Section 5 we demonstrate a prototypical implementation of our approach. Section 6 compares our ideas with related work and finally, Section 7 concludes this paper and outlines future work.

2 Preliminaries

Let us first review the lifecycle of processes that are enacted with the help of information systems. Fig. 1 depicts the lifecycle's four most prominent phases, c.f. [1].

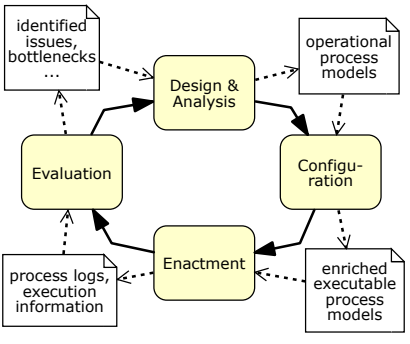


Fig. 1. BPM lifecycle

In the *design and analysis* phase, business processes are elicited and captured in graphical process models. These models need to be configured to be enacted on a process engine, which is done in the *configuration* phase. The resulting artifacts are executable process definitions. In the *enactment* phase, these process definitions are carried out by a process engine. Usually human interaction is embedded in the process and application services are called by the engine to integrate existing software systems. During *enactment*, process engines produce logs containing execution details of the process instances. The logs and runtime information can be analyzed in the *evaluation* phase. The cycle is re-iterated when the results of this phase are fed back into the analysis and design phase to improve the process.

Definition 1 (Process Model). *A process model P is a connected, directed graph (N, E) where $N = T \cup G \cup \{n_{in}, n_{out}\}$ is a finite set of nodes, with tasks T, gateways G, with $T \cap G = \emptyset$, and exactly one start and end event n_{in} and n_{out}, and $E \subseteq (N \setminus \{n_{out}\}) \times (N \setminus \{n_{in}\})$ is a set of edges connecting the nodes.*

In Definition 1, we capture process models as they are elicited in the *design and analysis* phase. They define activities and control flow structure. However, if enactment of such processes is desired, they need to be transformed into an executable format. This is done in the *configuration* phase. The business process execution language (BPEL) [3] is an industry standard that has clear execution semantics and can be executed by a process

engine, e.g., Apache ODE[1]. A common method is to translate graphical process models, e.g., defined as EPC [7] or BPMN [13] diagrams, into an executable representation, e.g., BPEL, cf. [21,14], and extend the generated code skeleton with execution details, as we describe next.

User tasks enable users to interact with the process and thus need a user interface. Usually these user interfaces are forms to display or enter data. Some activities in the process can be executed automatically, for instance, through running a script, calling an application or a Web service. This is realized through a service task that needs configuration to execute the corresponding logic. Both task types, user task and service task, may access data objects during the enactment of the process. Further, process control flow may diverge on databased exclusive gateways (XOR splits), depending on values of data objects of process instances. The technical engineer needs to assign the proper expressions at the XOR splits in the process model.

Definition 2 (Data Access). *Let D be the set of data objects within a process model P. A node can either read (r), optionally read (or), write (w), optionally write (ow), or not process (n) a data object. The* data access matrix *of nodes to data objects is a function: access* : $N \times D \rightarrow \{or, r, ow, w, n, (or, ow), (or, w), (r, ow), (r, w)\}$

Roles need to be configured for each user task defined in the process. If the concept of pools and lanes is used for role modeling, it is much easier for the technical engineer to configure the roles on the lane level. This solution greatly speeds up configuration of business processes with many user tasks. Likewise service tasks need to be configured to call a specific service.

Definition 3 (Assignment). *Let R be a set of roles that map to organizational functions and S a set of application services of that organization. The assignment of task $t \in T$ to a role $r \in R$ or a service $s \in S$ is a function: assign* : $T \rightarrow R \cup S$

The above steps yield a configured process model that can be executed on a process engine.

Definition 4 (Configured Process Model). *A configured process model $P_{conf} = (N, E, D, R, access, assign)$ is a process model, where each user task $T_{User} \subseteq T$ is assigned to a role and each service task $T_{Service} \subseteq T$ is assigned to a service, where $T_{User} \cup T_{Service} = T \bigwedge T_{User} \cap T_{Service} = \emptyset$. $G_{XOR} \subseteq G$ is the set of data-based exclusive gateways that define for each outgoing edge an expression based on data objects, which is captured by the access function*

3 Hardships in the Configuration of Processes

In traditional approaches, as we observe in many modern business process management systems, both model types from Definition 1 and 4 are decoupled. Business experts graphically design process models in the *design and analysis* phase and hand the results to the IT-department, where these models are transformed into executable formats (see above) and then need to be configured manually.

[1] Apache ODE – http://ode.apache.org

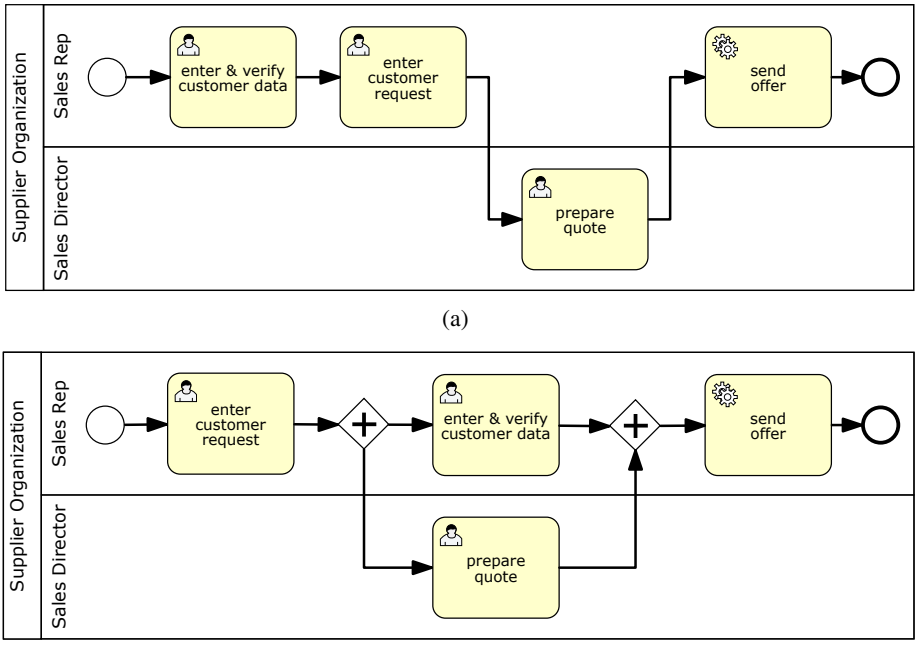


Fig. 2. Example process before (a) and after (b) redesign

Besides the error-proneness of this approach, this method has a major problem dealing with changes in the process model. If changes are made in the process model, it is a cumbersome task to keep the configured, executable process synchronized with the model; if changes are required on the technical model, they would need to be incorporated into the graphical model, which rarely happens, because again, these are manual tasks. The process models serve mainly for documentation purpose and quickly get out of sync with the reality.

Another hardship is to keep track of all data dependencies in a process model that need to be considered when configuring or redesigning a process model. While methods to ensure control flow consistency of process models where introduced more than a decade ago [2], dependencies between the data object access of nodes still expose a high risk of inconsistencies, and may require great effort to disclose.

Consider the example depicted in Fig. 2. Note that it is not obvious, which data dependencies exist in the model, i.e., upon changing the model with the intention of improving the process, one may unintendedly destroy the data flow consistency.

The example (Fig. 2(a)) shows a simplified process of a manufacturer company, where a sales representative stays in contact with a customer, who requested an offer. The sales representative enters the data of the customer, e.g., name and address, as well as the request of supplied goods. However, this person may not be authorized to create a quote, this is the task of the sales director for which he needs only information about the customer's request. After all information is set, the sales representative can send the offer to the customer.

Based on the assumption that the configuration of the process model in Fig. 2(a) is complete and error-free, to improve the runtime of this process, the tasks to enter the customer's data and request, and to create the quote could be conducted in parallel, as they are user tasks that may take a considerable amount of time. However, the task *prepare quote* requires the customer's request data, i.e., they have causal data dependencies. Any attempt to model them in parallel would lead to a missing data inconsistency. Since the data dependencies of this process model are hidden in technical details and not obvious, such violations wouldn't be disclosed until the process becomes implemented.

A better solution is depicted in Fig. 2(b) and could only be proposed if one has detailed insight into the data that is read and written by each task: Entering the customers data does not require the customer's request, nor does it require the prepared quote. From that, it follows that the sales representative could *enter the customer request* first, and while they *enter & verify the customer's data* the sales director can *prepare the quote*. Both concurrent paths modify distinct data objects and are synchronized before the offer is sent to the customer. Thus, no data dependencies are violated and no inconsistencies occur and the process is correctly redesigned.

In the next sections we present a set of redesign patterns that guide the improvement of business processes and the fundamentals of checking data flow consistency. By means of a prototypical implementation we show how we addressed the hardships identified above, i.e., manual configuration of technical details, poor support of data flow consistency checking, and the lack of roundtrip support in the business process life cycle.

4 Alleviating the Hardships

In the configuration phase of the business process lifecycle, tool support for changing business processes is most important. To avoid the above mentioned synchronization issues between process models and technical implementations, we merge them, i.e., we add technical details to the operational graphical models and generate executable process definitions of these models. While this concept is not new and quite common in model driven engineering [10], applied to process modeling it has to be altered in a domain specific way.

To reduce possible errors in the configuration phase, we designed a wizard guiding the technical engineer through these configuration steps. The first requirement to the wizard is that instead of typing names of resource assignments in the process models, the technical engineer should be able to select the resources from a list of available ones. Besides making sure, that only existing resources can be assigned, it is important for a process model to be executed correctly, that all data-dependencies in the process model are satisfied. We go into the inner mechanics of the data-dependency check in Section 4.2. But first, we discuss the impact of the consistency checker on business process redesign.

4.1 Supported Redesign Patterns

Support for process model changes, e.g., redesign, is not sufficiently available in current process aware information systems [12]. To better understand, what can be done

to support these changes, it is necessary to know what types of changes should be supported. Addressing this question, Reijers et al. identified 29 general redesign patterns for improving business processes [15]. Table 1 lists the patterns, that are reflected on the process model layer. These redesign patterns can only be applied correctly under certain preconditions regarding data flow. In the following, we define these preconditions and provide examples of how we support these redesign patterns.

Table 1. List of supported process model redesign patterns [15]

Redesign pattern	Support
operational patterns	
Order types	`possible`
Task elimination	`yes`
Triage	`yes`
Task-composition	`possible`
behavior patterns	
Resequencing	`yes`
Parallelism	`yes`
information patterns	
Control addition	`yes`
technology patterns	
Task automation	`yes`

Order types describes the problem of treating all instances in a process equally, even though some instances do not require all process steps.

pre: none

example: The subprocess "Packaging" of an order process is split into two different variants. One for single order positions and another one for multiple positions. The subprocess for single positions does not need the extra task "Choose delivery options", where the customer can choose whether to wait for all positions to be sent in one package.

requirements to the wizard: 1) The wizard could be extended to find violations of data consistency in process hierarchies. However, the link between parent and child model must be bidirectional for consistency checks, when changing a child model, that is referenced in many parents. 2) Dealing with variants can be implemented similarly, the only difference is that the link between the "parent" and "child" is horizontal instead of vertical.

Task elimination.

pre: A task $t \in T$ to be deleted may only write data $d \in D$ that is not read later in the process, or that was already available in the process. This includes data that contributes to the output of the process.

example: In a production process, there is a task "Check quality" at the end. It can be eliminated to save cost, since it does not produce data. The external behavior of the production process is still the same, however the non-functional property "quality" may have been reduced by this pattern.

support: The configuration wizard immediately gives feedback, if data produced by an eliminated task is required somewhere in the process.

Triage refers to (a) splitting a general task into several alternatives based on the instance data, to better utilize resources and make use of specialization. Also the contrary is possible, i.e., to (b) combine alternative tasks to a general task.

pre: (a) none (to split a task into alternatives)
(b) none (to join alternatives)

example: The task "Check order" can be split into "Check high volume order" and "Check low volume order". Assigning these tasks to different people in the organization may improve performance of the process through specialization.

support: The wizard supports copy-&-paste for tasks, allowing to split a task easily. After wiring the control flow and adding the data based routing gateway, the technical engineer needs to define the routing conditions of the gateway. The consistency checker immediately warns, if the routing conditions are invalid.

Task composition (a) combines small task into composite tasks or (b) divides large tasks into workable smaller tasks.

pre: (a) none (to split a task into several single steps)
(b) none (to merge tasks)

example: In a registration process, the two sequential tasks "Insert customer details" and "choose password" can be merged. The two respective input forms will be merged too and be assigned to the merged task.

requirements to the wizard: In order to properly support this redesign pattern, we plan to have a graphical editing of user forms used in the configuration wizard. To support task splitting, the user should be able to mark specific input fields of a selected task's form. The system then can split the form into two parts. Where one contains the marked input fields, and the other the remaining ones. The task itself will be duplicated and added in the control flow right behind the first task. Task merging would reverse this procedure, i.e., two sequential user tasks can be merged into one by combining the respective forms of each and assigning it to the resulting task.

Resequencing.

pre: Task *a* and *b* are in sequence, i.e., *b* is done after *a*. Task *b* may not read data that is produced only by *a* or later tasks in the process.

example: In the motivating example in Fig. 2(b), the tasks "enter customer request" and "enter & verify customer data" are resequenced, because they do not share any data dependency. However, they are conducted by the same person, which motivates to keep them in sequence.

support: Process modelers can switch tasks in sequence and get notified, whether there arise conflicts by doing so.

Parallelism.

pre: a) Additionally to the pre-condition of *Resequencing*, both tasks must not write the same data element. (to make sequential tasks parallel)
b) none (to sequentialize two parallel tasks)

example: In the motivating example in Fig. 2(b), the tasks "enter & verify customer data" and "prepare quote" are made parallel, because they do not access data produced by one another nor do they write to the same data object.
support: After rearranging the control flow in the process, the process modeler gets immediate feedback, if the data-dependencies still hold, and a successful execution of the process is possible.

Control addition means adding an extra quality control task to check the results of former tasks.
pre: none
example: After the "Approve credit" task performed by a clerk in a credit agency's core business process, a new "Verify credit approval" task is added. This new task will be performed by the controlling department, which can review the decision of the clerk and overrule it in case.
support: In the process modeler used by the wizard the task to be controlled can be copy-&-pasted. The new copy of the task needs to be assigned to another functional role that has the authorization to perform the quality control. Renaming the control task and connecting the nodes completes this pattern.

Task automation.
pre: The user task $u \in T_{user}$ to be automated should have an equivalent service implementation s configured in the system. Service s has to write at least the subset of written data from u, that is used in the process later on.
example: The "Send invoice" task of an order process is to be automated. This is done by manually sending an email, and will be done by a service afterwards.
support: Once the service is implemented and configured, the technical engineer has to change the type of the task to be automated from *user*- to *service*-task and select the existing implementation from the list of configured services.

4.2 Consistency Checking

As indicated earlier, checking consistency of data dependencies and making sure that no data anomalies will be manifested through execution is of utmost importance. For instance, if some task within a process reads a data object that is not yet initialized, this might lead to deadlocks in the process execution.

Data anomalies can be categorized into three basic categories: *missing data*, *redundant data*, *conflicting data* [17]. As indicated above, a missing data anomaly occurs in general when there is a chance for some task to be executed where it reads a data object that was never written (initialized) before. Redundant data anomaly occurs when a data object is written by some task but never read by any subsequent task. Finally conflicting data anomaly occurs when a data object is written by two or more tasks concurrently.

Sun et al. in [17] have given a comprehensive description of such anomalies and how to detect them, due to space limitations, we will discuss a core subset of such anomalies. Basically, we need two pieces of information to detect such anomalies: The read/write access taken by each node t against each data object d and the execution ordering, i.e., behavioral relationships, among nodes within the process model. In their work [17], the

Table 2. Data anomalies

Anomaly	Description
Missing data warning type 1	A task b reads an object that is optionally written by a preceding task a
Missing data warning type 2	A task b optionally reads an object that was never written by any preceding task
Missing data error	A task b reads an object that was never written by any preceding task
Redundant data warning	A data object is written by a task a but it was never read afterwards
Conflicting data warning	Two concurrent tasks a,b write a data object where at least one of them optionally writes the data object
Conflicting data error	Two concurrent tasks a,b write a data object

authors did not specify an exact algorithm to build the behavioral relationship among tasks. Therefore, we depend on the notion of behavioral profiles [20] to obtain the behavioral relationship among tasks within a process. A behavioral profile identifies the behavioral relationship between any pair of nodes within the model. This relationship is one of four values: (1) strict order $\rightsquigarrow$, (2) concurrent $\|$, (3) exclusive # or (4) inverse order $\leftsquigarrow$. Moreover, for each task, we need to know whether it is optional.

Definition 5 (Behavioral Profile). *Let N be the set of nodes within a business process model. The* behavioral profile *of a business process model is a function $bhp : N \times N \rightarrow \{\rightsquigarrow, \leftsquigarrow, \|, \#\}$ that assigns a behavioral property, strict order, inverse order, parallel, or exclusive, between each pair of nodes within the business process model.*

If two tasks a, b appear in strict order, $bhp(a,b) = \rightsquigarrow$, then task a executes before task b. Similarly, if two tasks are concurrent then they can be executed in any order. Exclusiveness means that at most one of the two tasks can execute within a process instance. A node $t \in N$ is *optional*, noted as $opt(t) = true$, if there is at least one other node $s \in N$ where $s \neq t$ and $bhp(s,t) = \#$. As the name indicates, the inverse order relation is the inverse of the strict order relation, $bhp(a,b) = \rightsquigarrow \Rightarrow bhp(b,a) = \leftsquigarrow$ and is defined for readability reasons only.

The other input to check consistency of data dependencies and lack of anomalies is the read/write relation among tasks and data objects. In Definition 2, it was shown that reading/writing a data object by a node can be optional. This, in turn, allows to generate finer grained levels of checks, e.g., warnings rather than errors. For instance, if a task b reads a data object d and another task a, where $bhp(a,b) = \rightsquigarrow$, optionally writes d then, the consistency checker will warn for a possible missing data anomaly. That is because, at execution time of b the data object d *might* not have been written by task a. On the other hand, if there was no task that writes d at all before b, an error is generated. Table 2 informally describes the different anomalies that can be identified with our approach. Also, it is possible to give hints to the user to help resolve such anomalies. For instance, for a missing data error, it could be possible that the task a that first writes a data object read by another task b executes after b, known as *late initialization* [17].

Based on definitions 5 and 4, we can define the different types of data anomalies: missing data, redundant data and conflicting data.

Definition 6 (Missing data Anomaly). *Let N be the set of nodes and D be the set of data objects within a business process model P_{conf} respectively. P_{conf} suffers from a* missing data anomaly *iff:*
$\exists t \in N\, \exists d \in D\, \nexists s \in N : \{r\} \in access(t,d) \wedge bhp(s,t) = \rightsquigarrow \wedge (\{w\} \in access(s,d) \vee \{ow\} \in access(s,d))$ *-Error.*
$\exists t \in N\, \exists d \in D\, \exists s \in N : \{r\} \in access(t,d) \wedge bhp(s,t) = \rightsquigarrow \wedge (opt(s) = true \vee \{ow\} \in access(s,d))$ *-Warning type 1.*
$\exists t \in N\, \exists d \in D\, \nexists s \in N : \{or\} \in access(t,d) \wedge bhp(s,t) = \rightsquigarrow \wedge (\{w\} \in access(s,d) \vee \{ow\} \in access(s,d))$ *-Warning type 2.*

Definition 7 (Redundant data Anomaly). *Let N be the set of nodes and D be the set of data objects within a business process model P_{conf} respectively. P_{conf} suffers from a* redundant data anomaly *iff:*
$\exists t \in N\, \exists d \in D\, \nexists s \in N : \{w\} \in access(t,d) \wedge bhp(t,s) = \rightsquigarrow \wedge (\{r\} \in access(s,d) \vee \{or\} \in access(s,d))$.

Definition 8 (Conflicting data Anomaly). *Let N be the set of nodes and D be the set of data objects within a business process model P_{conf} respectively. P_{conf} suffers from a* conflicting data anomaly *iff:*
$\exists t,s \in N\, \exists d \in D : \{w\} \in access(t,d) \wedge bhp(s,t) = \| \wedge \{w\} \in access(s,d)$ *-Error.*
$\exists t,s \in N\, \exists d \in D : \{w\} \in access(t,d) \wedge bhp(s,t) = \| \wedge (opt(s) = true \vee \{ow\} \in access(s,d))$ *-Warning .*

5 Prototypical Implementation

To validate the concepts discussed above, we implemented a tool that supports the aforementioned process model redesigns. It does not automatically identify deficiencies of a process model and apply the adequate pattern to improve it, but rather supports process experts, e.g., process engineers, to change the structure of the model according to these patterns and verify whether the improved model's data flow remains consistent.

This tool is based on Oryx[2]—an extensible process modeling platform that enables users to easily and efficiently create process models on the Web and allows developers to extend the tool's modeling capabilities. New modeling languages or extensions thereof can be added as stencil sets, new functionality can be made available to users through an elaborate plugin infrastructure [9]. Fig. 3 shows the example process model before redesign in Oryx.

As process definition language we resorted to a subset of BPMN 2.0 that suffices to capture most of today's process models, yet remains simple enough to be comprehended by the majority of users [11]. This subset allows to define tasks, to be conducted through software services or humans, control flow splits, either through exclusive choice or parallel branches, as well as pools and swim lanes to group activities according to the organizational role that conducts these tasks.

[2] The Oryx Research Project – http://oryx-project.org/research

To lift technical aspects of a process model implementation into the graphical model, we extended the BPMN 2.0 stencil set with properties that capture the configuration of roles, user tasks, system tasks, and XOR splits. Further, we implemented a wizard plugin that assists the process engineer to configure the process in three steps.

1. **Assign roles.** This is done through accessing a directory service, e.g., LDAP, and requesting the roles (user groups) of an organization. The technical engineer can choose for each lane, which group is responsible to conduct the tasks contained in the swim lane, cf. Definition 3.
2. **Configure tasks and control flow.** In this step, the main configuration takes place. Data forms are assigned to user tasks, services are registered with service tasks, and condition expressions of an XOR split are assigned. Data forms and services have been set up before, i.e., they are stored in a repository along with metadata that specifies input data, optional input data, and output data. By that metadata, the data consistency checker can derive, which data is read and which is written by the respective activity, cf. Definition 2.

   The wizard offers a configuration interface for each activity, where the engineer can select the data form or service, respectively. This is shown in Fig. 3. XOR splits are configured through writing expressions that use variables, or specifying at most

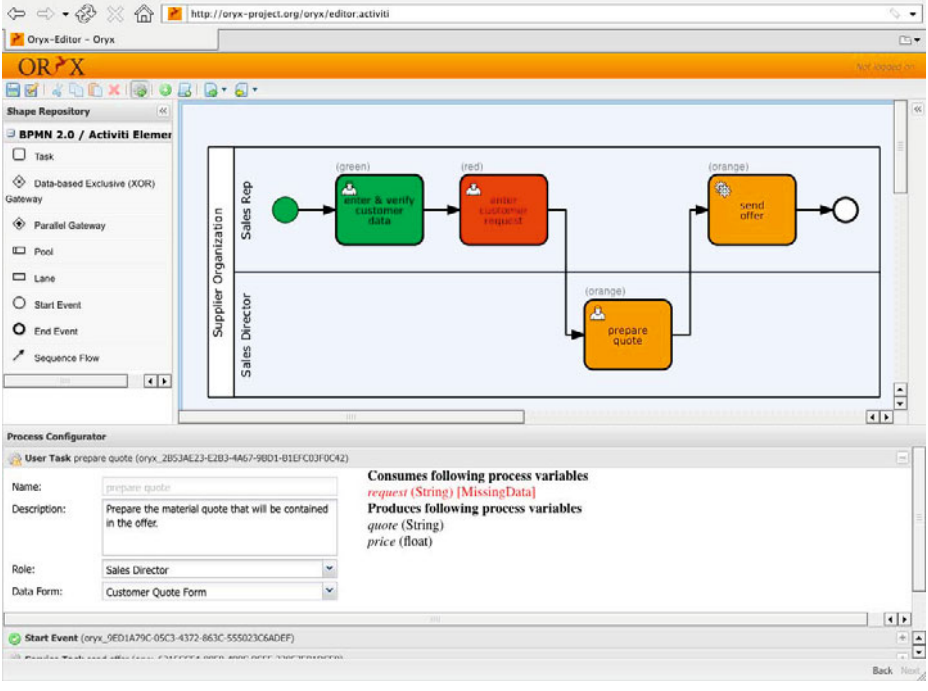


Fig. 3. Screenshot of the Configuration Wizard with an example business process model. The configuration wizard shows the user task “prepare quote”.

one edge as the default edge chosen when no other condition evaluates to true, cf. Definition 4.

3. **Deploy process.** We implemented a deployment service, that transforms the process from the stencil set model into a readily executable format and installs it on a running instance of the open source BPM platform Activiti[3].

In the second step *Configure tasks and control flow*, every change triggers a (non-blocking) consistency check: Data dependencies are computed from data access and control flow, cf. Section 4.2, by deriving the behavioral profile for the nodes.

Any violations are displayed in the process model in the following way. If an activity or XOR split has not been configured sufficiently yet, it is highlighted in red. If it has been configured and no violations have been detected, it is highlighted green. If a task or XOR split has been configured, but consistency violations have been detected, the corresponding element is highlighted orange, and in the configuration interface the variables that caused the violations are shown in red along with the type of violation. In Fig. 3, this is the case for the tasks *prepare quote* and *send offer*, because the task *enter customer request* has not been configured yet, and thus the data object *request* is not available. As soon as this task is configured correctly, the data flow within the process will be consistent. Then, the process can be deployed to an engine.

6 Related Work

The first problem in configuring business process models is how to deal with the gap between the graphical models and the implemented versions interpreted by a process engine. As this problem is crucial in the popular research field of *model driven engineering*, there exist many conceptual solutions already. The most direct way to bridge the gap between models and implementation is to make models executable, by adding the necessary details [10]. We use the very same concept and apply it to configuring BPMN 2.0 models with necessary execution artifacts. Besides this method, also other techniques such as round-trip engineering were applied to process models [22]. This approach establishes associations between business workflow models and source code. These can be used, when it is not possible to integrate all information into one model.

The second problem is the efficient support for changes in graph-based models. In this area two simple refactorings for UML activity diagrams were introduced in [4], i.e., making actions parallel and sequentializing parallel actions. More recently 11 concrete business process model refactorings were defined in the context of large process repositories [19]. Tool support is still missing though, and these particular refactorings are mainly applicable only to process trees or process models with variants. For graph-based models, even automatic refactoring methods were proposed in [18]. These refactorings rearrange control-flow in the model, such that they get well-structured, i.e., can be split into hierarchical singe-entry single-exit regions.

Similarly to our continuous consistency checking for data dependencies, the authors in [8] present continous validation while modeling, but restrict the checks to structural properties, i.e., soundness of process models.

[3] Activiti BPM Platform – `http://www.activiti.org/`

Related to the *data dependency check* we presented in this paper is [16], where the authors identified a set of data anomalies as: redundant data, lost data, missing data, mismatched data, inconsistent data, misdirected data, and insufficient data. However, the paper just signaled these types of anomalies without an approach to detect them. A refinement of the aforementioned set of data anomalies was given in [17]. Although the authors in [17] provide algorithms to check such data anomalies, they do not discuss how to build the behavioral relationship within a model. In comparison to our work, we depend on the notion of behavioral profiles [20], which can be computed efficiently, to get the behavioral relationships among tasks within a process model. Then, we build a CRUD matrix to identify relationships among tasks and data items. Moreover, we provide a finer grained level of feedback. That is, we distinguish between warnings and errors. Recently, the authors in [6] presented an algorithm for efficient detection of data anomalies in business processes. They also provide detailed feedback and produce errors and warnings. However, they require the models to be structured to be able to analyse them.

Van Hee et al. [5] present a case study on how consistency between models of different aspects of a system can be achieved. They model object life cycles derived from CRUD matrices as workflow nets and later synchronized with the control flow. Their approach, however, does not present strategies how inconsistencies between data flow and control flow can be removed. Compared to our approach, we do not require the knowledge of object life cycles before hand. Rather, we extract it from the model under investigation.

7 Conclusion

Supporting changes in business processes is a major requirement in industry, that is not yet sufficiently solved. In this paper, we addressed this problem by augmenting the configuration and (re-)design phases of the BPM lifecycle with a configuration wizard relying on a model driven approach. The configuration wizard is integrated in the open-source modeling platform Oryx and builds on BPMN 2.0 process models. Continuous consistency checks ensure that data-related modeling errors are detected immediately, saving a lot of effort to detect potential errors manually. Thus, changes in the process models are facilitated and encouraged.

We further identified seven redesign patterns in [15] and explained how they are supported by the wizard. We introduced a fine grained definition of data-anomalies that can occur in business processes and explained how consistency checks can detect them. Finally, we compared our solution with existing work in these areas.

Future work includes providing support for the two redesign patterns we do not support yet, i.e., *Order types* and *Task composition*. Also, we want to integrate a recommender system based on the consistency checking results. Cases of *late instanciation*, mentioned in Section 4.2, that refer to read access to a process variable, that is written later in the process can be detected and pointed out to the technical engineer. Other suggestions, along the lines of *"task A reads data d, which is missing in the process, but provided by form f (or service s)"* can also be provided to the technical engineer configuring the process.

As we mentioned in Section 5, the current approach is limited to a subset of the most commonly used BPMN 2.0 modeling constructs, cf. [11]. As engines will support more advanced process constructs, such as event-based gateways, timer events, etc, we shall extend the wizard and data consistency checker with them.

References

1. van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M.: Business process management: A survey. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 1–12. Springer, Heidelberg (2003)
2. van der Aalst, W.M.P.: Workflow verification: Finding control-flow errors using petri-net-based techniques. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, pp. 161–183. Springer, Heidelberg (2000)
3. Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guízar, A., Kartha, N., et al.: Web Services Business Process Execution Language Version 2.0. OASIS (2007)
4. Boger, M., Sturm, T., Fragemann, P.: Refactoring browser for UML. In: Liu, Y., Awasthi, P., Unland, R. (eds.) NODe 2002. LNCS, vol. 2591, pp. 366–377. Springer, Heidelberg (2003)
5. Hee, K.M.v., Sidorova, N., Somers, L.J., Voorhoeve, M.: Consistency in model integration. Data Knowl. Eng. 56(1), 4–22 (2006)
6. Heinze, T.S., Amme, W., Moser, S.: Effiziente Abschätzung von Datenflussfehlern in strukturierten Geschäftsprozessen. In: CEUR Workshop Proc., vol. 705 (2011)
7. Keller, G., Nüttgens, M., Scheer, A.: Semantische Prozeßmodellierung auf der Grundlage "Ereignisgesteuerter Prozeßketten (EPK)". Inst. für Wirtschaftsinformatik (1992)
8. Kühne, S., Kern, H., Gruhn, V., Laue, R.: Business Process Modelling with Continuous Validation. In: Workshops of BPM. LNBIP, vol. 17, Part 3. Springer, Heidelberg (2008)
9. Decker, G., Overdick, H., Weske, M.: Oryx – sharing conceptual models on the web. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 536–537. Springer, Heidelberg (2008)
10. Mellor, S., Balcer, M.: Executable UML: A foundation for model-driven architectures. Addison-Wesley, Reading (2002)
11. zur Muehlen, M., Recker, J.C.: How much language is enough? Theoretical and practical use of the business process modeling notation. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 465–479. Springer, Heidelberg (2008)
12. Mutschler, B., Reichert, M., Bumiller, J.: Unleashing the Effectiveness of Process-Oriented Information Systems: Problem Analysis, Critical Success Factors, and Implications. IEEE Trans. Syst., Man, Cybern. 38(3), 280–291 (2008)
13. OMG: Business Process Model and Notation (BPMN) 2.0 Specification (January 2011)
14. Ouvans, C., Dumas, M., ter Hofstede, A.H.M., van der Aalst, W.M.P.: From BPMN process models to BPEL web services. In: ICWS 2006, pp. 285–292. IEEE, Los Alamitos (2006)
15. Reijers, H., Liman Mansar, S.: Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. Omega 33(4), 283–306 (2005)
16. Sadiq, S., Orlowska, M., Sadiq, W., Foulger, C.: Data flow and validation in workflow modeling. In: ADC, pp. 207–214 (2004)
17. Sun, S.X., Zhao, J.L., Nunamaker, J.F., Sheng, O.R.L.: Formulating the data-flow perspective for business process management. Info. Sys. Research 17(4), 374–391 (2006)
18. Vanhatalo, J., Völzer, H., Leymann, F., Moser, S.: Automatic workflow graph refactoring and completion. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 100–115. Springer, Heidelberg (2008)

19. Weber, B., Reichert, M.: Refactoring process models in large process repositories. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 124–139. Springer, Heidelberg (2008)
20. Weidlich, M., Polyvyanyy, A., Mendling, J., Weske, M.: Efficient computation of causal behavioural profiles using structural decomposition. In: Lilius, J., Penczek, W. (eds.) PETRI NETS 2010. LNCS, vol. 6128, pp. 63–83. Springer, Heidelberg (2010)
21. Ziemann, J., Mendling, J.: EPC-based modelling of BPEL processes: a pragmatic transformation approach. In: MITIP 2005. Citeseer (2005)
22. Zou, Y., Lau, T., Kontogiannis, K., Tong, T., McKegney, R.: Model-driven business process recovery. In: WCRE 2004, pp. 224–233. IEEE, Los Alamitos (2004)

Systematic Derivation of Class Diagrams from Communication-Oriented Business Process Models*

Arturo González[1,3], Sergio España[2,3], Marcela Ruiz[2,3], and Óscar Pastor[2,3]

[1] DSIC, [2] Centro de Investigación ProS, [3] Universitat Politècnica de València
agdelrio@dsic.upv.es
{sergio.espana,lruiz,opastor}@pros.upv.es

Abstract. Enterprise information systems can be developed following a model-driven paradigm. This way, models that represent the organisational work practice are used to produce models that represent the information system. Current software development methods are starting to provide guidelines for the construction of conceptual models, taking as input requirements models. This paper proposes the integration of two methods: Communication Analysis (a communication-oriented requirements engineering method [1]) and the OO-Method (a model-driven object-oriented software development method [2]). For this purpose, a systematic technique for deriving class diagrams from business process models is proposed. The business process specifications (which include message structures) are processed in order to obtain class diagram views, which are integrated to create the class diagram incrementally. Then, using the *OLIVANOVA* framework, software source code can be generated automatically. The paper also discusses the advantages and current limitations of the technique. Results show that, although there is room for improvement, the technique is feasible and it does facilitate the creation of the class diagram.

Keywords: Information systems, requirements model, business process model, model transformation, class diagram, Communication Analysis, OO-Method.

1 Introduction

Requirements Engineering (RE) for enterprise information systems (ISs) mainly deals with engineering business process models and requirements models. These artefacts serve several purposes. For instance, they are used to specify the needs of an organisation with regards to communication and information processing. This way, they serve as input for later software development activities such as conceptual modelling and information system design. Since the business strategy and the information technology are expected to be aligned [3], data models must be designed so that they ensure the IS support to business processes.

* Research supported by projects GVA ORCA (PROMETEO/2009/015), MICINN PROS-Req (TIN2010-19130-C02-02), the MEC FPU grant (AP2006-02323), and co-financed with ERDF.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 246–260, 2011.

This work is part of a long-term research effort to obtain a model-driven IS development method that covers from business process modelling (BPM) and RE to software-code generation. For this purpose, the paper proposes the integration of two methods: Communication Analysis, a communication-oriented BPM and RE method [1], and the OO-Method, a model-driven object-oriented software development method [2]. The OO-Method conceptual model consists of four complementary models that cover the structural (Object Model), behavioural (Functional Model and Dynamic Model) and interactive (Presentation Model) aspects of the software system under construction. The derivation of the Dynamic Model (a set of UML State Machine diagrams) is discussed in [4]. This paper makes the following contributions:

- A systematic technique for deriving the Object Model (an extended UML Class Diagram) from a Communication Analysis requirements model is presented. The business process specifications are processed in order to obtain class diagram views (which include classes, attributes, services, and structural relationships, as well as many of their properties). Message Structures, a technique for the specification of communicative interactions, plays a primary role in the derivation. The class diagram is created incrementally, by means of view integration.
- The technique is illustrated by means of a running example that is fully described in [5]. Other ongoing validations are also discussed. Current results of using the technique prove its feasibility and also allow identifying some improvement issues.

We are conscious that, in recent years, many attempts have been made to provide transformations from requirements models to class diagram-like models. We argue that our approach goes one step further in its ability to address complex information systems (covering both structural and dynamic aspects) and its potential[1] to be embedded in a fully-fledged model-driven development framework such as *OLIVANOVA*, which supports code generation [6].

The paper is structured as follows. Section 2 reviews related works. Section 3 describes a case that is later used for illustration purposes. Section 4 summarises the two methods that are integrated using the derivation technique. Section 5 overviews the proposed technique. Section 6 describes the derivation technique and illustrates it with a running example. Section 7 discusses the advantages, limitations and risks of using the technique. Section 8 presents conclusions and future work.

2 Related Works

Previous works position Communication Analysis in the BPM and RE arena (e.g. [1]); in short, this method shares the communicational orientation of the Language-Action Perspective [7]. Since this paper presents the derivation of class diagrams, we focus the review of the literature to works that confront the problem of reasoning data-oriented models from function- or process-oriented models.

[1] At this moment, the derivation technique is intended to be applied manually. We do not analyse herein the possible automation of the rules; this is planned as future work.

Some approaches start from a use case model and provide guidance for creating a class diagram, either by directly applying linguistic patterns [8], by bridging the gap with intermediate models such as sequence diagrams [9] or activity graphs [10], or by extending use case models [11] or extended task descriptions [12] with information flow specifications. This paper goes further in providing a systematic approach that incrementally obtains the class diagram. The concept of class diagram view allows for a step-wise derivation (see Section 5) and could be adopted by previous approaches; also, approaches that define precedence relations among use cases or tasks could also adopt the ordering procedure (see Section 6). Moreover, the selection of the BPM method to be integrated with a conceptual modelling method is justified, industrial experience and previous experiments show that Communication Analysis models have more completeness and fewer modularity errors than Use Case models [13].

Some approaches advocate taking a data-centric perspective to business process modelling [14-16]. These approaches consider the data that flows among information processing tasks, whereas we advocate taking a communicational perspective that implies considering message flows between information systems and actors. The process-centred modelling approach presented in this paper aims to fit the model-driven development paradigm and this is facilitated by predefining the order of model creation steps. However, the consistency guidelines that the above-mentioned works propose can be useful to check the results of applying our derivation technique.

With regards to view integration, it has been acknowledged as a research challenge since the advent of data-model analysis and design. Many authors have tackled this issue, mainly with the intention of bringing agreement among different stakeholders [17]. However, these works do not attempt to derive a complete conceptual model from a complete requirements model, but deal with obtaining different partial conceptual model views and then merging them, solving eventual structural conflicts.

3 Description of the Illustration Case: SuperStationery Co

In order to illustrate the proposal, we use a running example about a fictional intermediary company that provides office material to its clients. The reader should bear in mind that the case is only intended to illustrate the derivation rules; due to deliberate simplifications, neither the requirements nor the conceptual model cover all the needs of a company of such kind. The case is described in full detail in [5]; in this paper, we focus on part of the sales management (acronym: SALE) business process.

To place an order, most clients phone the Sales Department, where they are attended by a salesman. Then the client requests products that are to be sent to one or many destinations (a client company can have decentralised offices). The salesman takes note of the order. Then the Sales Manager assigns the order to one of the many suppliers that work with the company. The supplier can either accept or reject the order. In case of acceptance, the salesman sends a copy of the order to the Transport and lnsurance Departments, where the corresponding clerks arrange the logistics and prepare the insurance documentation, respectively. The supplier reports to the company when the truck picks up the goods from the supplier warehouse.

4 Summary of the Two Methods Being Integrated

4.1 Communication Analysis

Communication Analysis is a BPM and RE method that proposes undertaking IS analysis from a communicational perspective [1]. It is has been successfully applied in big projects; e.g. the integration of Anecoop S. Coop (a Spanish major fruit-and-vegetables distributor) with its circa 100 associated cooperatives.

Communication Analysis offers a requirements structure and several modelling techniques for BPM and requirements specification. Among these techniques, the Communicative Event Diagram and the Event Specification Templates are primary for conceptual model derivation. The *Communicative Event Diagram* is intended to describe business processes from a communicational perspective. A *communicative event* is a set of actions related to information (acquisition, storage, processing, retrieval, and/or distribution), that are carried out in a complete and uninterrupted way, on the occasion of an external stimulus. Business process model modularity is guided by unity criteria [18]. For each event, the actors involved are identified, as well as the corresponding communicative interactions and the precedence relations among events. Fig. 1 shows a communicative event diagram of the running example.

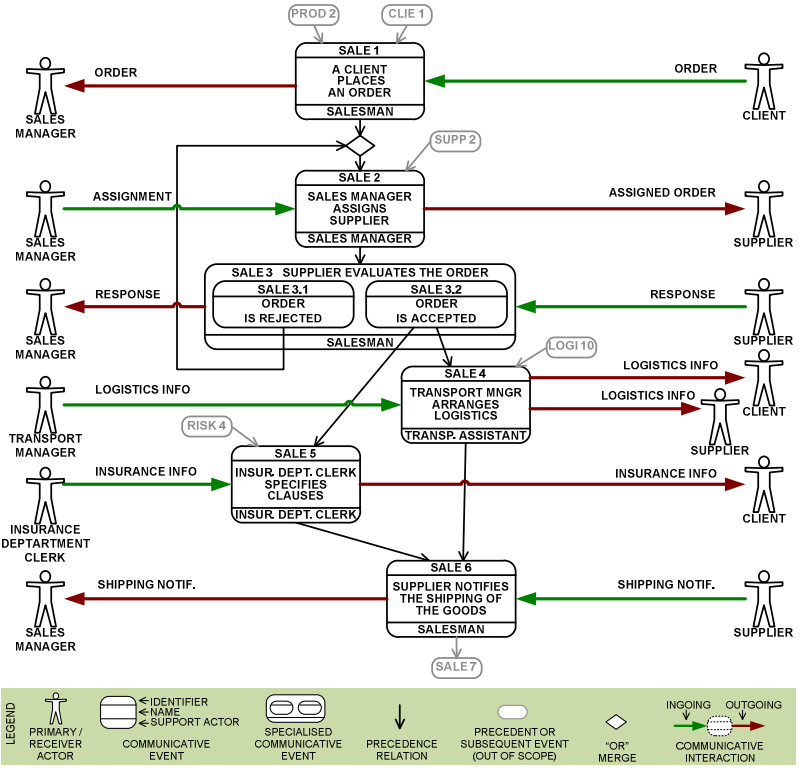


Fig. 1. Communicative event diagram of (part of) the *Sales management* business process

Table 1. Message structure of communicative event SALE 1 in analysis time

FIELD	OP	DOMAIN	EXAMPLE VALUE
ORDER =			
< Order number +	g	number	10352
Request date +	i	date	31-08-2009
Payment type +	i	text	Cash
Client +	i	Client	56746163-R, John Papiro Jr.
DESTINATIONS =			
{ DESTINATION =			
< Address +	i	Client address	Blvd. Blue mountain, 35-14A, 2363 Toontown
Person in charge +	i	text	Brayden Hitchcock
LINES =			
{ LINE =			
< Product +	i	Product	ST39455, Rounded scissors (cebra) box-100
Price +	i	money	25,40 €
Quantity >	i	number	35
}			
>			
}			
>			

LEGEND For more details see [19]

TYPES OF FIELDS
- DATA FIELDS HAVE A BASIC DOMAIN (e.g. Request date)
- REFERENCE FIELDS HAVE A BUSINESS OBJECT TYPE AS A DOMAIN (e.g. Client)

GRAMMATICAL CONSTRUCTS

< + > AGGREGATION SUBSTRUCTURE { } ITERATION SUBSTRUCTURE [|] SPECIALISATION SUBSTRUCTURE

INFORMATION ACQUISITION OPERATIONS

g GENERATION i INPUT d DERIVATION

The *Event Specification Template* allows structuring the requirements associated to a communicative event. Among other requirements, it contains a description of the new meaningful information that is conveyed to the IS in the event. This is specified by means of *Message Structures*, a modelling technique that is based on structured text. Previous work [19] presents the grammar of Message Structures and provides guidelines for their application during analysis and design (they are used differently in each development stage). Table 1 shows the message structure of communicative event SALE 1 in analysis time. To create the message structure, the analyst had to interview the users and to analyse the available business forms. Note that it merely describes the message and it is, therefore, an analysis artefact. The structure of message fields lies vertically and field properties can be arranged horizontally; e.g. information acquisition operation, field domain, an example value provided by users.

4.2 OO-Method

The OO-Method is a model-driven object-oriented software development method that focuses on modelling enterprise ISs at the conceptual level [2]. The *OLIVANOVA* framework supports conceptual modelling and, by means of a *Model Compiler*, the software code is generated automatically [6]. This feature adds extra value to deriving a conceptual model from a requirements model, the essential part of the presented work. The OO-Method conceptual model consists of four complementary models:

- *Object Model*. An extended UML Class Diagram that specifies the IS memory.
- *Dynamic Model*. A collection of UML State Machine Diagrams that represents the valid lifecycles of objects and the interactions among them.
- *Functional Model*. It describes the reaction of the IS using an abstract pseudocode.
- *Presentation Model*. It models the user interface by means of abstract patterns.

The derivation technique presented herein is intended to produce the Object Model, which is the core model of the OO-Method. The derivation of the Dynamic Model is presented in [4] and ongoing research is addressing the other models. The Object Model includes classes that contain attributes and services (which can be atomic services or transactions). Classes are interrelated by means of structural relationships that have restrictions such as maximum and minimum cardinalities for each role.

5 Overview of the Derivation Approach

We first present an overview of the derivation strategy (see Fig. 2); more detailed guidelines are presented in Section 6, along with the running example. Both the Communicative Event Diagram and the Event Specification Templates contain relevant information for the derivation of the Object Model.

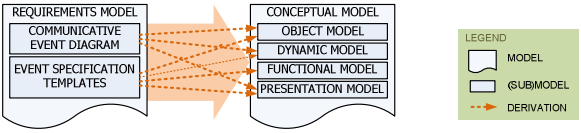


Fig. 2. Derivation strategy from Communication Analysis to OO-Method models

The main step consists of processing the message structures in order to derive the class diagram view that corresponds to each communicative event. We refer as *class diagram view* to a portion of the class diagram that includes one or several portions of classes and structural relationships among them. By portion of a class we mean some or all of the attributes and services of the class (including their properties). The concept of class diagram view with regard to conceptual models is analogous to the concept of relational view with regard to relational database schemas [17].

Each communicative event derives a different class diagram view. The derived views are integrated to obtain the complete class diagram, the same way that different relational views are integrated to obtain a single database schema [17]. This integration can be approached in two ways [5]: post-process view integration or incremental view integration; we opt for the second. The procedure consists of the following steps. First, the communicative events are sorted. Then, the events are processed in order, one by one. The class diagram view that results from processing each communicative event is integrated into the class diagram under construction. This way, new classes and structural relationships are added to the class diagram, as well as class attributes and services. Fig. 3 exemplifies this approach. Note that one communicative event can affect one or several classes (event IV), and one class can be affected by one or many communicative events (class C).

By processing the communicative events in a predefined order and by performing an incremental view integration, we intend to prevent a situation in which a newly-derived class diagram view refers to a class diagram element that has not yet been derived (e.g. trying to add an attribute to a yet inexistent class). In any case, if the requirements model suffers incompleteness, this situation cannot be fully prevented.

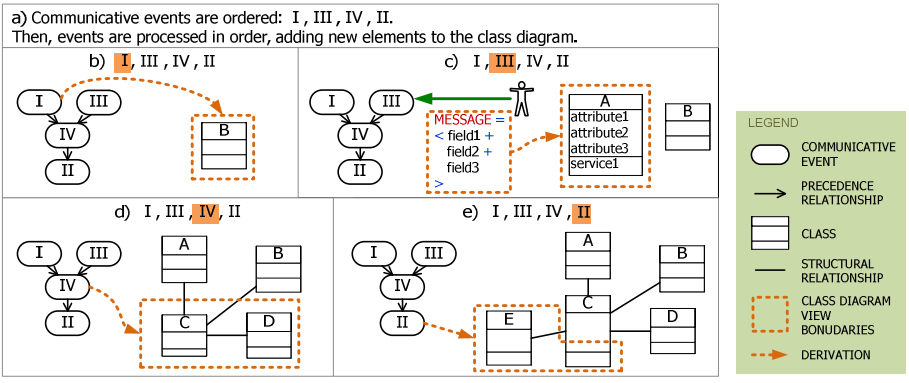


Fig. 3. Simplified example of incremental class diagram view integration

6 Derivation Guidelines and Illustration

In the following, the derivation guidelines are presented as rules (rule numbers between parentheses are used for cross-reference), along with the running example.
First, the communicative event diagram is extended with precedent communicative events so as to obtain a rooted directed graph (R1), where the start node is designated as the root. For this purpose, we need to include any communicative event that is not included in the diagram but precedes a communicative event which is included in the diagram (this step is repeated until no more events are included). In such cases, the diagrams are connected by means of *out-of-scope* precedence relations (see Fig. 1).

As a pre-processing step, the analyst marks those reference fields which indicate that (part of) a business object is being extended with new information (R2). The marks can be supported by means of a new Boolean property for reference fields (i.e. a new column in the message structure template) named *Extends business object.* Only one reference field per aggregation substructure can be marked. These marks will facilitate the derivation process later on.

Then, the communicative events depicted in the extended communicative event diagram are ordered according to the precedence relations (R3). For this purpose, a partially ordered set of communicative events is obtained by removing the loopbacks that might appear in the extended communicative event diagram. The remaining precedence relations define a strict partial order among the events. Any known procedure for topological sorting can be used for obtaining the ordered list of events. Any linear extension is suitable for derivation purposes and the choice is not expected to influence efficiency; for instance: SUPP 2, PROD 2, CLIE 1, SALE 1, SALE 2, SALE 3, LOGI 10, SALE 4, RISK 4, SALE 5, SALE 6. The events are processed in this order.

In the following, the processing of SALE 1 and SALE 2 is explained. Since SUPP 2 and PROD 2 have already been processed, the class diagram is under construction and it already contains elements. The complete derivation is described in [5].

In order to derive the class diagram view that corresponds to SALE 1, the message structure of this communicative event is processed (see Table 1). It consists of an initial aggregation structure named ORDER that includes an iteration substructure

named DESTINATIONS, which in turn includes another iteration substructure named LINES. Both iteration substructures have an aggregation substructure inside of them (DESTINATION and LINE respectively). Each aggregation substructure not containing a reference field that has been marked according to R2, leads to the derivation of a new class (R4). That is, whenever an aggregation structure is not specifying the provision of new information about an already known business object, a new class is created. This way, the three aggregation substructures lead to the derivation of the classes CLIENTORDER, DESTINATION and LINE, respectively. We now focus on the first.

The name of the aggregation substructure can be used to name the class (R5) (that is, the class could have been named ORDER); however, the analyst can decide to give the class a different name. For each data field of the substructure, the newly-derived class is added an attribute (R6). For instance, the data field Order number leads to adding the attribute order_number to the class CLIENTORDER[2]. Fig. 4 shows the derived classes and attributes. Most of the attribute properties (see Table 2) are derived from the event specification templates; in the absence of the necessary requirements, the analyst should ask the users or decide relying on his/her own judgement.

Table 2. Specification of attributes of class CLIENTORDER

Attr. name	Id	Attr. type	Data type	Size	Requested	Null allowed
order_number	yes	Constant	Autonumeric	-	yes	no
request_date	no	Constant	Date	-	yes	no
payment_type	no	Variable	String	30	yes	yes

The names of the attributes are derived from the names of the data fields (R7) (spaces are replaced by underscores and lowercase letters are used).

If a contextual restriction in the event specification template specifies how organisational actors identify business objects, then this information is used to select which attributes constitute the class identifier (R8). For instance, the requirements model states that "orders are identified by an order number," so the attribute order_number is designated the class identifier (column Id in Table 2). In the absence of such restriction, an artificial autonumeric identifier can be created.

The attribute type should be asked to the users[3]. By default, all identifier attributes are defined as constant; the rest are defined as variable (R9) (column Attribute type).

Attribute data types are derived from data field domains (R10). Communication Analysis prescribes a few basic domains for data fields (which serve the purpose of clarifying the meaning of the messages), whereas the OO-Method offers a wider selection of data types for attributes (code generation is intended). The correspondences between both are defined in [5]. For instance, the data type of request_date is Date because the field domain of Request date is date).

In a newly derived class, all the attributes are set as requested at creation time (R11). CLIENTORDER does appear for the first time during the processing of SALE 1.

[2] As explained below, reference fields do not derive attributes but structural relationships.

[3] Once a *constant* attribute has been initialised, its value cannot be modified. In contrast, *variable* attributes can be modified (as long as the proper class services are defined) [2].

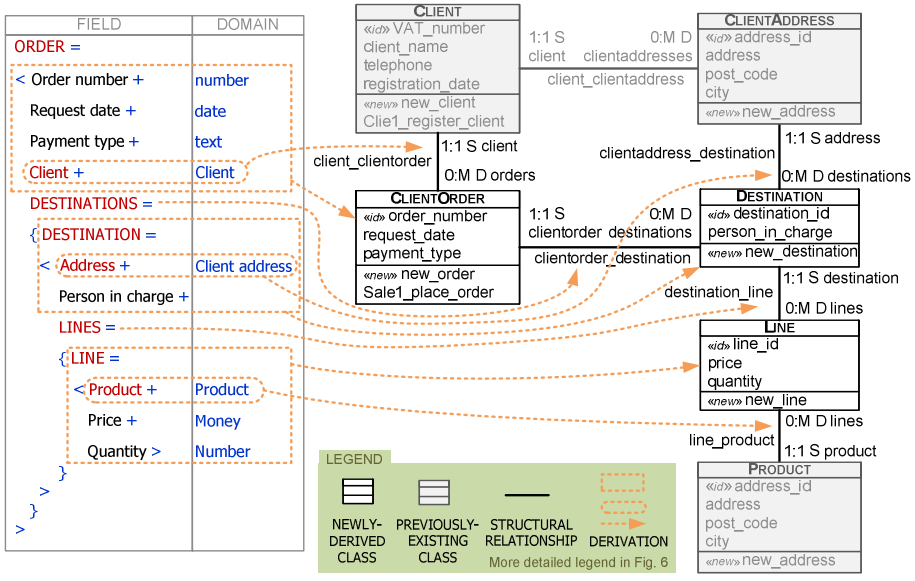


Fig. 4. Derivation of the class diagram view that corresponds to communicative event SALE 1

Whether an attribute should allow null values or not needs to be asked to the users. However, by default, all attributes that are part of the class identifier are added a restriction so that they do not allow null values; the rest of the attributes are set to allow null values (R12) (see column Null values in Table 2).

The nesting of complex substructures leads to the derivation of structural relationships between the classes that correspond to the substructures (R13)[4]. For instance, since the substructure DESTINATION is part of the substructure ORDER, then the corresponding classes (CLIENTORDER and DESTINATION, respectively) are related by means of a structural relationship. If the nested substructure is an aggregation substructure then the maximum cardinality[5] on the side of the referenced class is 1; if it there is an iteration substructure in between, then this cardinality in M (many) (R14). Since DESTINATIONS (the substructure that relates ORDER and DESTINATION) is an iteration substructure, then the structural relationship has maximum cardinality M on the side of class DESTINATION (i.e. the iteration specifies that one client order can have many destinations). The rest of the cardinalities depend on structural restrictions that appear in the event description templates (R15). A similar reasoning applies to order lines. See the resulting structural relationships and cardinalities in Fig. 4.

Reference fields lead to the derivation of structural relationships between the class that corresponds to the complex substructure that contains the reference field and the class that corresponds to the reference field (R16). The reference field named Client

[4] In *OLIVANOVA* the kind of relationship (i.e. association, aggregation, composition) does not have an impact on the generated code, so we opted for only using association relationships.

[5] We acknowledge that cardinality is often referred to as multiplicity since UML became a de-facto standard; however, OO-Method still uses this term [2]. Also, the notation in *OLIVANOVA* departs from the UML: CLASSA minA:maxA --- minB:maxB CLASSB.

leads to adding a structural relationship between the class being edited and the class that was derived from the message structure of communicative event CLIE 1. Salesman registers a client (see [5]). Thus, a structural relationship is created between the classes CLIENTORDER and CLIENT. The maximum cardinality on the role side of the referenced class is 1 (R17). Since the client is specified as a reference field, only one client can be associated to the order (otherwise it would be specified as an iteration). The rest of the cardinalities depend on structural restrictions (see R15). See the resulting structural relationship and its cardinalities in Fig. 4.

Additionally, a creation service is added to each newly-derived class (R18); for instance, a service named new_order is added to the class CLIENTORDER. For each data field, a data-valued inbound argument is added to the creation service (R19). For instance, the argument p_atrorder_number corresponds to the data field Order number and to the attribute order_number. For each reference field an object-valued inbound argument is added (R20). For instance, argument p_agrClient is added to the service; this attribute defines which client places the order. The properties of the inbound arguments coincide with their corresponding attribute properties (see Table 3).

Table 3. Specification of inbound arguments of service new_order

Argument name[6]	Data type	Size	Null allowed
p_atrorder_number	Autonumeric	-	no
p_atrrequest_date	Date	-	no
p_atrpayment_type	String	30	yes
p_agrClient	Client	-	no

When the class diagram view corresponds to a complex business object, then apart from the creation service another service is added to the main class; it triggers the IS reaction to the communicative event (R21). Therefore, a service named Sale1_place_order is added to the class CLIENTORDER. This service is triggered by the salesperson whenever s/he has finished introducing the information if the order; that is, after introducing the destinations and the lines. Only after this service is executed, the order is considered to be placed. As prescribed by the OO-Method, every service that is not a creation service needs an inbound argument that represents the instance of the class for which the service is invoked (R22) (see Table 4).

Table 4. Specification of the inbound argument of service Sale1_place_order[7]

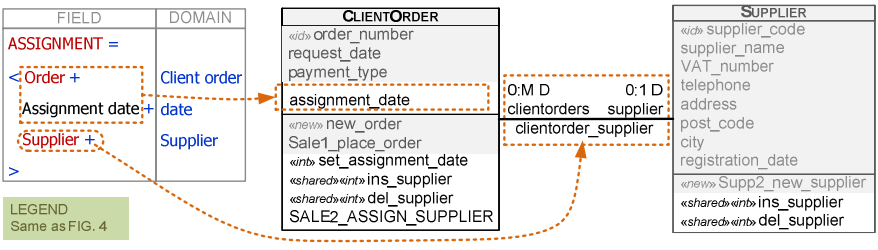
Argument name	Data type	Size	Null allowed
p_atrorder_number	Autonumeric	-	no

After an order is placed, the Sales Manager assigns the order to one of the many suppliers that work with SuperStationery. Thus, communicative event SALE 2 affects the same business object as communicative event SALE 1; namely, the client order. A reference field marked as extending a business object (see R2) leads to extending an

[6] Prefixes in the argument names follow *OLIVANOVA* naming conventions.

[7] It is possible to identify which of the many services of a complex business object actually triggers the IS reaction by prefixing its name with the identifier of the communicative event.

existing class (R23). For instance, the reference field Order in the message structure of SALE 2 indicates that the client order is affected and, therefore, the class CLIENTORDER is extended (see Fig. 5). The data fields in the message structure of Sale 2 lead to adding new attributes to this class (see R6), whereas the reference fields lead to adding new structural relationships between the class CLIENTORDER and other classes that already exist in the class diagram under construction (see R16).



Attr. name	Id	Attr. type	Data type	Size	Requested	Null allowed
assignment_date	no	Variable	Date		no	yes

Fig. 5. Class diagram view of SALE 2 and specification of the new attribute

With regards to data fields, the field Assignment date leads to adding an attribute named assignment_date to the class CLIENTORDER. Fig. 5 specifies the details of the new attribute. All attributes that are added to a class as a result of a class extension have the following properties: these attributes are not part of the identification function, the attribute type is Variable, they are not requested upon creation, and they allow nulls (R24). The data type is derived from the domain of the field (see R10).

With regards to reference fields, the field Supplier references a business object that appears in the communicative event SUPP 2. A structural relationship is defined between the classes CLIENTORDER and SUPPLIER. The maximum cardinality in the side of SUPPLIER is 1 (see R17); the minimum is 0 because orders are not assigned to suppliers when they are placed, but it occurs in a later moment in time.

If new attributes are added to an extended class, then a service is added in order to set their values (R25). Therefore, a service named set_assignment_date is added to the class. Furthermore, given the cardinality of the structural relationship, two shared services are included in both classes (this is prescribed by the OO-Method [2]); namely an insertion shared service named ins_supplier and a deletion shared service named del_supplier. When the reaction to a communicative event requires the execution of several services of the same class, then a transaction is created in order to ensure their atomic execution (R26). For instance, SALE2_ASSIGN_SUPPLIER is defined in order to execute atomically the events set_assignment_date and ins_supplier.

Fig. 6 shows the class diagram (Object Model) that results from processing the remaining events. The Dynamic Model, whose derivation is explained in [4], specifies system behaviour, so attributes such as order_state are not needed for this purpose.

A strength of our approach is that it is integrated in a model-driven development framework. By applying a model compilation with *OLIVANOVA*, the software source code can be automatically generated (see some snapshots of the application in [5]).

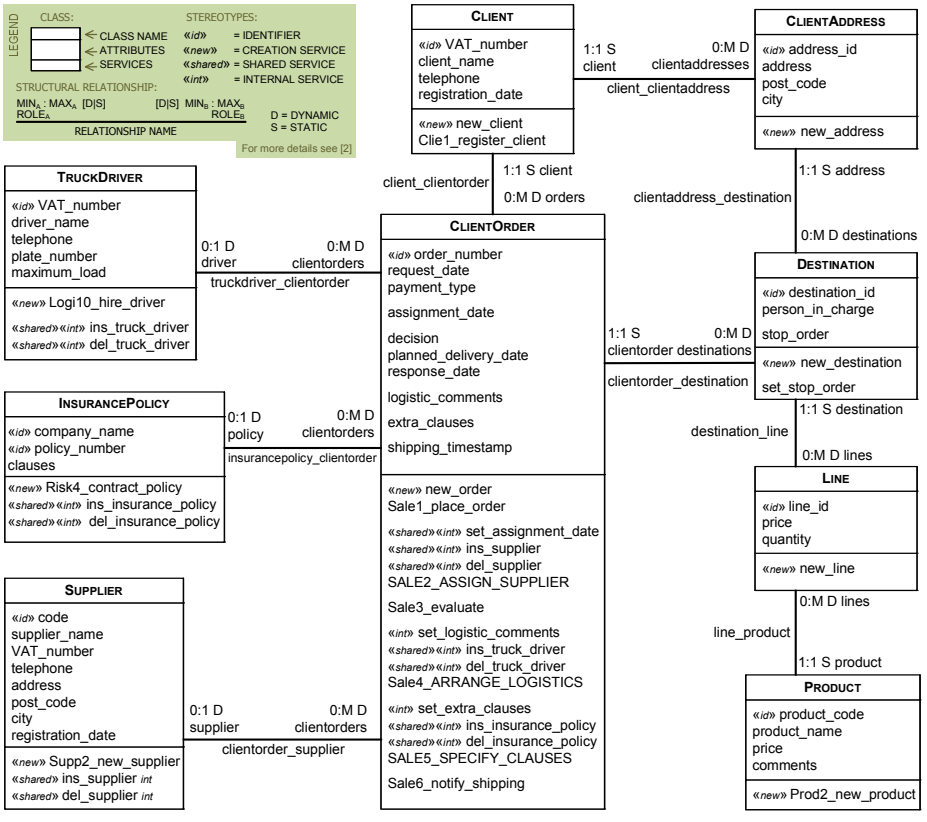


Fig. 6. Class diagram that corresponds to the SALES business process of SuperStationery Co

7 Discussion

The researchers have so far applied the derivation approach to create three conceptual models (the biggest has a size of 537 IFPUG function points). The most detailed *lab demo* up to now is the SuperStationery Co. case [5]. The experience has proved that the approach is feasible in practice. By following this systematic procedure to create conceptual models we were able to ground most conceptual modelling decisions in the requirements model elements (diagrammatic or textual). However, we acknowledge that we need to investigate how requirements-model incompleteness and invalidity affect the outcome. During our lab demos, the analysts simply took some decisions *on the fly* in order to deal with missing requirements. Since we intend to automate derivation as much as possible, we should at least ensure that missing requirements do not induce invalidity in the conceptual model. It may prove necessary to verify requirements models before applying the derivation rules.

The derivation technique allows tracing class-diagram elements back to elements of the requirements model. With the proper support, this information can be valuable.

Also, two experiments have been carried out to validate the technique. A pilot experiment was operated in University of Twente in 2009, involving 3 students during 3 months (an optional bachelor course was set up to train them). No quantitative analysis was possible, but interesting qualitative conclusions were drawn, what led to improving the derivation guidelines. A controlled experiment was operated in Valencia in 2010, involving 29 last-year master students during 3 months. 14 students were trained in the derivation technique presented in this paper and 15 in conceptual modelling from a textual description of the organisational work practice (as OO-Method and *OLIVANOVA* practitioners use to do in real projects). During 7 hours over 4 days the students created the conceptual model of a photography agency IS. In both experiments the derivation technique was applied manually but conceptual modelling was supported by *OLIVANOVA*. We are currently analysing the quality of their class diagrams using a set of metrics based on [20] and [21] in order to compare both groups, so no quantitative conclusions are available yet.

These experiences with the derivation technique have allowed us to identify issues that require further investigation. In the following, we summarise some of them[8].

- A more efficient support for modelling and deriving basic create/update/delete operations is convenient. We face the challenge of formalising the specification of basic CRUD operations in a way that they do not increase the workload of the analyst, but they can still be processed during the conceptual model derivation.
- Service derivation can be improved. The OO-Method Functional Model offers an expressive pseudocode to specify class service algorithms. The derivation technique is currently being extended to guide the modelling of valuations (rules that specify the behaviour of atomic services) and transaction formulas.
- It is possible to specify initialisation values for message structure fields in design time [19]. For data fields, this property could be used to determine attribute default values. For reference fields, this property could be used to define a transaction that initialises the link of a structural relationship.
- There is no provision for dealing with specialised message structures in the derivation guidelines. We plan to investigate how to profit from using class inheritance relations.
- We are currently deriving the class diagram from scratch, disregarding a scenario in which part of the class diagram could have already been created.
- Although Communication Analysis and the OO-Method are successfully applied in industrial-sized projects separately, the scalability of the integrated derivation technique needs to be assessed.

8 Conclusions and Future Work

Enterprise information systems (IS) must support organisational work practice. Thus, the memory of the IS (typically specified by a data model) needs to be aligned with the business processes (typically specified by a business process model).

This paper presents a technique for deriving data models from business process models. Specifically, the paper focuses on deriving a OO-Method Object Model (a

[8] A more in depth discussion of issues that require further investigation can be found in [5].

UML Class Diagram) from Communication Analysis requirements models (i.e. a communicational perspective on business process modelling). The technique offers a systematic way of tackling the derivation problem by sorting the communicative events and processing them in order. For each communicative event, a class diagram view is derived. This is achieved by processing the message structure that corresponds to the event (i.e. a specification of the message conveyed by the actor to the IS, in the form of structured text). Each class diagram view constitutes a portion of the class diagram, so the class diagram is incrementally constructed by integrating the views. The derivation of the Dynamic Model (which specifies e.g. the different states of a client order) is addressed in [4]. Both derivations provide enough information (structural and behavioural aspects of the system) to the conceptual model to allow for automatic code generation with *OLIVANOVA*.

The technique has been put in practice in several *lab demos* and tested with last-year master students. Results show that the approach is feasible and promising. There is also room for improvement. The technique at its current state facilitates deriving the structure of interrelated classes, the attributes and services of the classes. Future work includes analysing experimental data and providing guidelines to derive the Functional Model and the Presentation Model (an abstract interface specification).

This work is part of a research effort to provide a model-driven information system development method that covers from requirements engineering to automatic software-code generation. Thus, we are developing a tool that supports Communication Analysis requirements modelling, using the Eclipse Modelling Framework [22]. We are also implementing transformation rules that are intended to automate conceptual model derivation as much as possible, using the Atlas Transformation Language. Moreover, further validations are planned, including an action research application of the derivation technique in a pilot development project.

Last but not least, we acknowledge that practitioners are usually reluctant to use non-standard notations. We therefore plan adopt the Business Process Modeling Notation (BPMN) [23] to support Communication Analysis BPM. The BPMN Choreography Diagram is the first candidate, but a careful investigation needs to be carried out to adopt the notation while preserving the concepts and criteria of the method. Then we intend to adapt the derivation technique to deal with the new BPM notation.

Acknowledgments. We thank anonymous reviewers for their helpful comments.

References

1. España, S., González, A., Pastor, Ó.: Communication Analysis: a requirements engineering method for information systems. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 530–545. Springer, Heidelberg (2009)
2. Pastor, O., Molina, J.C.: Model-Driven Architecture in practice: a software production environment based on conceptual modeling. Springer, New York (2007)
3. Henderson, J.C., Venkatraman, N.: Strategic alignment: leveraging information technology for transforming organizations. IBM Syst. J. 38(2-3), 472–484 (1999)
4. España, S., Ruiz, M., Pastor, Ó., González, A.: Systematic derivation of state machines from communication-oriented business process models. In: RCIS 2011. IEEE, Los Alamitos (2011)

5. España, S., González, A., Pastor, Ó., Ruiz, M.: Integration of Communication Analysis and the OO-Method: Manual derivation of the conceptual model. The SuperStationery Co. lab demo. Technical report ProS-TR-2011-01 (2011), http://arxiv.org/pdf/1101.0105
6. CARE Technologies. *OLIVANOVA* The Programming Machine, http://www.care-t.com
7. Weigand, H.: Two decades of the language-action perspective. Introduction. Commun. ACM 49(5), 44–46 (2006)
8. Díaz, I., Sánchez, J., Matteo, A.: Conceptual modeling based on transformation linguistic patterns. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, pp. 192–208. Springer, Heidelberg (2005)
9. Insfrán, E., Pastor, Ó., Wieringa, R.: Requirements engineering-based conceptual modelling. Requir. Eng. 7(2), 61–72 (2002)
10. Kösters, G., Six, H.-W., Winter, M.: Coupling use cases and class models as a means for validation and verification of requirements specifications. Requir. Eng. 6(1), 3–17 (2001)
11. Fortuna, M., Werner, C., Borges, M.: Info Cases: integrating use cases and domain models. In: 16th International Requirements Engineering Conference, pp. 81–84. IEEE Press, Los Alamitos (2008)
12. de la Vara, J.L., Sánchez, J.: System modeling from extended task descriptions. In: 22nd Int. Conference on Software Engineering and Knowledge Engineering, pp. 425–429 (2010)
13. España, S., Condori-Fernandez, N., González, A., Pastor, Ó.: An empirical comparative evaluation of requirements engineering methods. J. Braz. Comp. Soc. 16(1), 3–19 (2010)
14. Nigam, A., Caswell, N.S.: Business artifacts: An approach to operational specification. IBM Syst. J. 42(3), 428–445 (2003)
15. Reijers, H.A., Limam, S., Van Der Aalst, W.M.P.: Product-based workflow design. J. Manage. Inform. Syst. 20(1), 229–262 (2003)
16. Sun, S.X., Zhao, J.L., Nunamaker, J.F., Liu Sheng, O.R.: Formulating the data-flow perspective for business process management. Inform. Syst. Res. 17(4), 374–391 (2006)
17. Batini, C., Lenzerini, M., Navathe, S.B.: A comparative analysis of methodologies for database schema integration. ACM Comput. Surv. 18(4), 323–364 (1986)
18. González, A., España, S., Pastor, Ó.: Unity criteria for business process modelling: A theoretical argumentation for a Software Engineering recurrent problem. In: RCIS 2009, pp. 173–182. IEEE, Los Alamitos (2009)
19. González, A., Ruiz, M., España, S., Pastor, Ó.: Message Structures: a modelling technique for information systems analysis and design. In: WER 2011 (2011), Extended version available http://arxiv.org/abs/1101.5341
20. Lindland, O.I., Sindre, G., Sølvberg, A.: Understanding quality in conceptual modeling. IEEE Softw. 11(2), 42–49 (1994)
21. Moody, D.L.: The Method Evaluation Model: A theoretical model for validating information systems design methods. In: ECIS 2003 (2003)
22. Ruiz, M., S. España, A. Gonzalez, and O. Pastor, Análisis de Comunicaciones como un enfoque de requisitos para el desarrollo dirigido por modelos. In: Avila-García, O., Cabot, J., Muñoz, J., Romero, J.R., Vallecillo, A. (eds.) DSDM 2010, JISBD, pp. 70-77 (2010)
23. OMG: Business Process Modeling Notation (BPMN) version 2.0 (2011), http://www.omg.org/spec/BPMN/2.0/

Verification of Timed BPEL 2.0 Models*

Elie Fares, Jean-Paul Bodeveix, and Mamoun Filali

IRIT, Université de Toulouse
firstname.lastname@irit.fr

Abstract. Web services are increasingly becoming a major part of our daily lives. Many web services composition languages have been developed to describe the way a group of distributed web services interact with each other. In this matter, BPEL is one of the highly used composition languages. In this work, we are interested in verifying BPEL processes. Several works have addressed this issue before, but to our knowledge, a formalism that captures both the behavioral and the timing aspects of all the constructs of BPEL 2.0 does not exist. In this paper, we introduce a verification framework for timed BPEL models. We show how the relative and the absolute time of BPEL can be treated. We also give examples of temporal and timed properties that are supported in our framework. The verification is based on a transformation of all the BPEL constructs to the process algebra language, FIACRE.

Keywords: Web services, BPEL, formal methods, verification.

1 Introduction

Web services are distributed applications which are designed to achieve a specific business task over the web. In order to carry out the business goals, these web services should be composed to interact together. The interaction is done by means of exchanging messages over public interfaces described in XML-based languages such as the Web Services Description Language WSDL [17]. Orchestration is one of the mechanisms that addresses service composition. WS-BPEL (Web Services Business Process Execution Language) [3] is a well known service composition language addressing orchestration. It defines business processes through the orchestration of different partners interactions. However, BPEL lacks a formal semantics and is defined informally in natural language. Thus, the validity of user requirements cannot be verified over BPEL processes before the actual implementation of the system occurs. This weakness of BPEL can be mended by formal methods. In fact, giving formal semantics to BPEL constructs provide a basis to reason formally over BPEL processes. Thanks to such a feature, the validity of user requirements can be checked before the actual implementation takes place. Several works have addressed this issue (see section 6). However, we believe that a unique formalism that captures both the behavioral and time

* This work has been partially sponsored by the french ANR project ITEMIS and Aerospace Valley project TOPCASED.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 261–275, 2011.

aspects of BPEL 2.0 is still missing. The contribution of this work lies in the covering of all the constructs of BPEL without doing any abstractions on its time aspects. This is done by mapping all of the BPEL constructs including time related constructs to a process algebra language, FIACRE. Furthermore, we will explain the method on how to deal with both the relative (`For duration`) and the absolute time (`Until deadline`) of BPEL.

The rest of the paper is organized as follows. In Section 2, we give an overview of the BPEL and the FIACRE languages. Moving on to Section 3, we introduce the idea of the transformation from BPEL to FIACRE bringing in the FIACRE patterns for the BPEL's basic activities and the scope structured activity. We wrap up this section by explaining how the BPEL timed constructs are treated in FIACRE. In Section 4, we present our verification framework. We then illustrate our technique through an example in Section 5. Moreover, in Section 6, we present the related works concerning the formal verification of BPEL. Finally, we draw a conclusion of the paper and list an overview of our future works.

2 A Brief Overview of BPEL and FIACRE

2.1 BPEL

BPEL [3] is a language that describes the behavior of a business process by interacting with a group of partner web services. This interaction, which is done for web services orchestration purposes, is supported thanks to `Partnerlinks`. `Partnerlinks` represent the static part of a BPEL process and are described in a WSDL document where the operations offered by a web service are also given. The dynamic part of BPEL is desribed by means of activities.

Basic Activities define the elementary operations of the business process. They are the usual ones such as `Receive`, `Reply`, `Invoke`, `Assign`, `Throw or Rethrow`, `Exit`, `Empty`, `Validate` to type check the value of a variable, `Wait` to delay the execution and the less usual ones such as `Compensate` and `CompensateScope` to trigger the so-called compensation handlers.

Structured Activities define the order in which nested activities are executed. These are the `Sequence` and the `Flow` activities for the sequential and parallel execution respectively, the `While`, the `RepeatUntil`, the `If`, the `Pick` for a choice over message events or alarm events and the `ForEach`. Finally, the `Scope` (see section 3.2) activity may be associated to a fault handler for internal faults handling, a compensation handler for undoing successfully finished scopes, a termination handler that controls the forced termination of a scope and an event handler that manages message and alarm events.

Moreover, links may be used inside a `Flow` activity to provide additional control over the order of execution of parallel activities. Each activity may have multiple outgoing links as well as multiple incoming links. Every outgoing link has a transition condition that is associated with it. The transition conditions are boolean expressions based on the process data which are written in other languages such as XPath [4]. These transition conditions are evaluated by the targeted activities in the form of a JoinCondion.

Timed Constructs use relative time within the `For Duration` primitive and absolute time within the `Until deadline` primitive. These features are used by a `wait` activity or an `<on alarm/>` event of a pick activity or of an event handler.

2.2 FIACRE

FIACRE [7] is a formal intermediate language dedicated to the modeling of both the behavioral and timing aspects of systems. It is used for formal verification and simulation purposes. Basically, FIACRE is a process algebra where hierarchical components communicate either through synchronous messages by means of ports or through shared memory by means of variables. These ports and variables define the interface of a component. These components are leaves (**process**) or nodes (**component**).

- **process** : describes a sequential behavior using symbolic transition systems based on [9]. Accordingly, a process is defined by a set of states and transitions. Each transition has a guard and a non deterministic action that may synchronize on a timed port and may update local or shared variables.
- **component** : describes the parallel composition of subcomponents. Moreover, the components may introduce variables and ports shared by its subcomponents. Real time constraints and priorities can be attached to ports. The real time constraint attached to a port `p` is a time interval `I`. This means that once the event `p` is enabled, the execution should wait at least the minimum bound of `I` and at most its maximum bound.

In the example given in Fig. 1, two process instances of `P` are composed together by synchronizing on the ports `ping, pong` and by sharing the variable `w`. We show the textual syntax and the graphical syntax where processes are illustrated as automatons and components as communicating boxes through ports (•) or shared variables (■). We will use both notations in the rest of the paper.

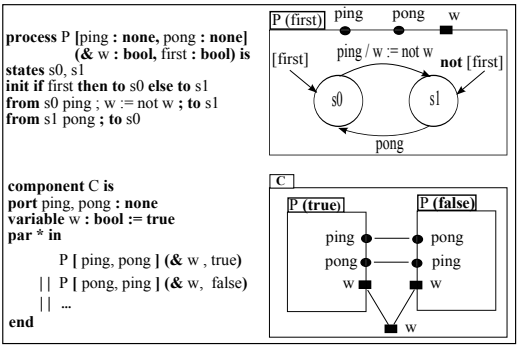


Fig. 1. Graphical notations

3 From BPEL to FIACRE

Based on the structure of a BPEL process, the static part of BPEL (WSDL) is modeled in FIACRE as global types. As for the dynamic part consisting of the primary activity of the BPEL process and its associated handlers, it is modeled as an outermost FIACRE component containing the FIACRE pattern of the composition of primary activity with these handlers (Fig: 2).

This outermost component builds a parallel composition of component instances representing the BPEL nested activities. Moreover, BPEL basic activities are transformed to FIACRE processes while BPEL structured activities and handlers – being able to contain other activities – are transformed to FIACRE components.

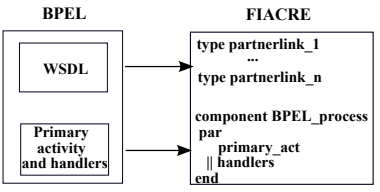


Fig. 2. Transformation structure

3.1 Modeling the WSDL

The interaction with the environment is supported by Partnerlinks. In BPEL, each Partnerlink may contain two roles (`myRole` and `partn-erRole`) typed with `Porttype`. Each of the `Porttype` declares several operations used to receive (Input) or send (Output) messages. Consequently, this structure is modeled in FIACRE by creating two different enumerated types named `inputs` and `outputs` used to model respectively the inputs and the outputs of each operation. The type `inputs` (resp. `outputs`) will be the union of the type of :

- the `input` (resp. `output`) arguments of operations of the `myRole` of every Partnerlink.
- the `output` (resp. `input`) arguments of operations of the `partnerRole` of every PartnerLink.

Note that for abstraction purposes, only boolean and enumerated types are preserved during this transformation. Actually, with respect to model checking purposes, considering the actual values is not realistic because of state explosion.

3.2 Behavioral Aspects in FIACRE

Each BPEL activity is mapped to a FIACRE component sharing a common interface. It will consist of the set of FIACRE ports and shared variables that each pattern should have in order to enable their composition. We will start by introducing a basic interface (Fig. 3) containing 2 ports : S and F (start,finish) used to connect the activities in the composition. At the end of every activity, the finish port is synchronized with the start port of another. We will extend the interface progressively in the rest of the paper.

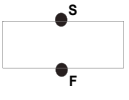


Fig. 3. Common interface

Common Behavior of Activities in FIACRE: All of the activities in FIACRE share a common behavior (Fig. 4). This behavior consists of modeling the outgoing and incoming links with their respective conditions (behavior with `suppJoinFailure = yes` [3]). Moreover, each activity will include additional control events and shared variables used in FIACRE for modeling the forced termination of activities or for reinitializing the activity in case it is nested inside of a repeatable construct. In Fig. 4, we have used some notations for clarity

purposes, namely the transitions are labeled by symbolic names which are explained later. The outgoing transition `all-i` means that it can be made by every state enclosed in the box. The activity body square represents a specific FIACRE pattern depending on the type of the modeled activity. This could be any basic or structured activity (see for example Fig. 5).

Composing the activities: every activity waits for its start signal by synchronizing on its `S` port of Fig. 3 (`i-s` and `i-j`). In the same way, at the end of the activity, it signals its finish by synchronizing on the `F` port of Fig. 3 (`l-i` and `j-i`) used as the start of the subsequent activity.

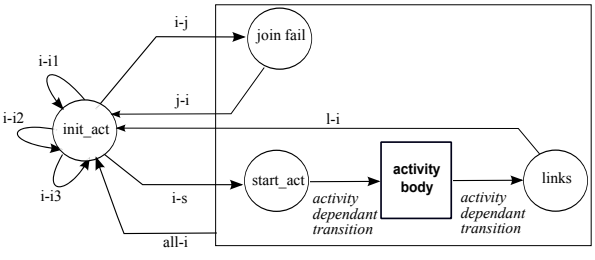


Fig. 4. Common behavior with suppJoinFailure = yes

Links Modeling : in order to model the BPEL links, the common interface is incremented by three variables. These variables are an integer variable `counter` initialized to the number of incoming links of an activity, a boolean variable associated with each of the outgoing links of an activity that corresponds to the transition condition of BPEL and another boolean variable `skip`. A source activity signals its end by decrementing the counter associated to its targeted activities by 1 and by updating its respective transition conditions (`l-i`). The targeted activities are executed once their counter evaluates to 0 and their join conditions evaluates to *True* (`i-s`). Nevertheless, if the join condition evaluates to *False* (`i-j`), the activity should set its outgoing links to False before terminating (`j-i`). Furthermore, in case the activity in question is a structured activity, all its nested activities should also set their outgoing links to False based on the Dead Path Elimination rule[1]. This is modeled by valuating the `skip` variable to *True* (`j-i`) which is evaluated before the start of each activity (`i-i3`).

Termination Modeling : in order to model the termination of activities in FIACRE, we add a boolean variable `stop` and a port `stpd` to the common interface which signal the termination demand and the occurrence of this termination respectively. Once the `stop` variable evaluates to True, the activity responds by communicating its `stpd` port and by transiting back to its initial state (transitions `all-i` and `i-i2`). On the contrary to some other techniques [10,14], we consider here a strong termination. This means that whenever a stop is demanded, the activities are no longer permitted to execute. This can be done in

[1] The dead path elimination is the result of two cases : (i) The join condition of an activity evaluates to False. (ii) If during the execution of a structured activity A, its specification says that an activity B nested within A will not be executed.

FIACRE by means of priorities or by adding explicitly the guard `on not stop` to each transition. This decision of termination is closer to the semantics of BPEL forced termination [3].

Reinitialization Modeling : Again, we increment the common interface by a shared variable `reinit` and a port `reinited` used to model the reinitialization of activities. The `reinit` variable is set to *True* by an enclosing repeatable construct. In this case, the activity will reset all its control variables namely its links counter variables, transition conditions and its skip variable before synchronizing on its `reinited` port.

Basic Activities. In order to model the basic activities, we increment the interface by the ports (`I_rec, O_rec, I_inv and O_inv`) used to communicate with the environment, the shared variable `vars` that represents the BPEL variables, the port `Fa` used to throw BPEL faults, the `exit` port and the shared variable `comp` used to trigger the compensation. We give in the following table Fig. 5 the FIACRE patterns for some of the interesting BPEL basic activities. For clarification purposes, we do not show all of the details of the modeling. For example, if a reply activity is not matched to a receive activity, then a default fault is thrown in BPEL. In the transformation of the basic activities, such details are explicitly modeled. In the following, we only explain the receive activity. In Fig. 5, the `receive` activity waits for a synchronization on the port `I_rec` of type `Inputs`. Once the message is received, it is either stored in the corresponding variable `var` or a `F_variableError` (BPEL default fault) is thrown. Because we have abstracted data values, the choice between storing the message or throwing the fault is non-deterministic.

Structured Activities and BPEL Handlers. Structured activities specify the order in which their nested activities are executed. The idea of the modeling in FIACRE consists of adding a Controller process which has no equivalence in BPEL. Based on the type of the BPEL structured activity, a different controller is created. This will specify the way the nested components of structured activities

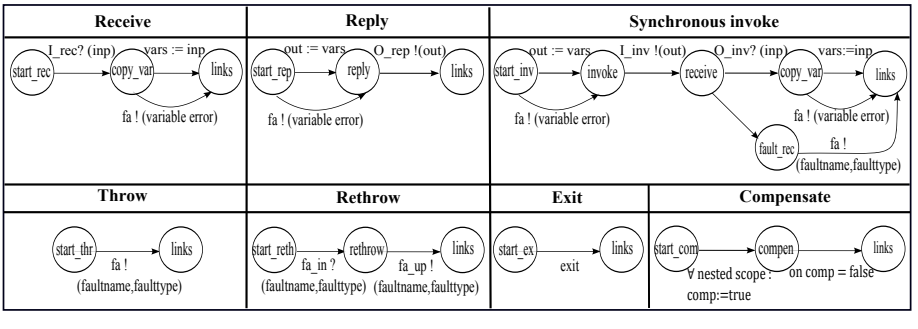


Fig. 5. BPEL Basic activities in FIACRE

are executed (sequential, parallel, conditional...). Moreover, it will capture the incoming and outgoing links of the structured activity. The components of all of the structured activities in FIACRE consist of a composition of a controller process with their nested activities. Due to the lack of space, we will not be able to display these patterns. However, we will present the pattern of the BPEL scope activity since it is the most complex construct.

Scope Component. The scope component in FIACRE is given in Fig. 6. It contains four subcomponents that respectively define nominal execution, fault and stop handling, termination handling and compensation handling. These components communicate with each other through local ports or variables, and they communicate with the scope environment through the scope component interface.

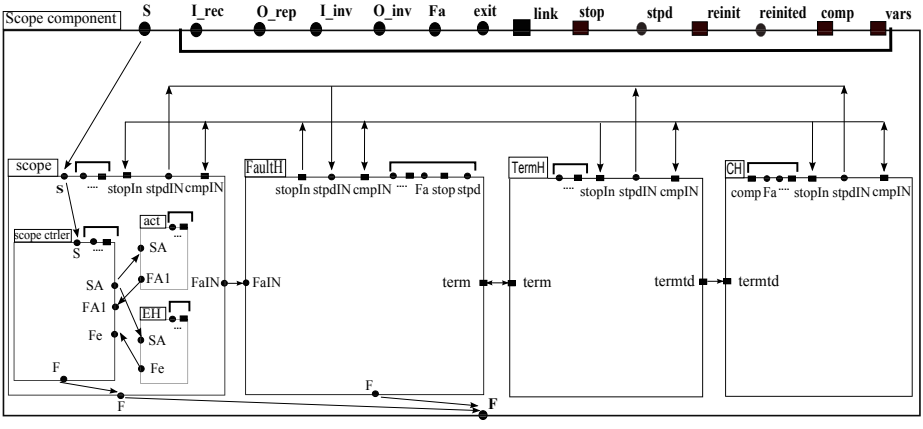


Fig. 6. Scope Component

Nominal Execution : It designates the execution of the first subcomponent of the scope. The start signal (port S) of the scope component is intercepted by the its scope controller `(scope ctrler)` leading to the execution of the primary activity of the scope and its associated event handler `(port SA)`. The end of both of these constructs leads to the end of the scope component `(F)`.

Fault and stop handling : it designates the execution of the fault handler associated with the scope. The fault handler in FIACRE is executed as a result of two scenarios :

1. Fault thrown inside the scope : this is the result of a synchronization on the port `Fa_In`. If the fault can be handled by the fault handler, it will start by asking the termination of the scope by valuating the `StopIn` variable to True. Then, it will wait until all of the activities synchronize on `stpdIn`. After that, the fault handler will proceed by executing its activities before eventually

synchronizing on the finish port `F`. If the fault could not be handled, it will be propagated directly to the enclosing scope by synchronizing on the `Fa` port shared with the interface of the scope component.
2. Forced termination signaled by the enclosing scope : the `Stop` variable shared with the scope component is valuated to True. As a result, the fault handler terminates the activities of the scope with the same mechanism described earlier and executes its termination handler (see termination handler) before signaling its termination to the enclosing scope by synchronizing on the `stpd` port shared with the scope component interface.

Termination Handler: The termination handler is only executed when an enclosed scope is forced to terminate by an enclosing scope. In FIACRE, this is modeled by the variable `term` that is valuated to True by the scope's fault handler whenever a forced termination is demanded. In addition, the scope should be in running mode in order for it to be terminated. In FIACRE, these conditions are checked by the termination controller (embedded in the TH component). If these conditions are fulfilled, then the termination handler is executed. All the faults thrown inside of the termination handler are treated internally and are never propagated. This is why no fault ports appear in the interface of the termination handler. We note that in our encoding, the innermost-first termination order of BPEL is respected. It is so because the `term` variable is reset to False only after the execution of the termination handler.

Compensation Handler: The compensation handler is triggered by a compensate or compensateScope activity embedded in one of the Fault-Compensation-Termination handlers of the enclosed scope. In FIACRE, this is modeled by valuating the `comp` variable to True. Furthermore, the compensation handler of a scope is only executed if the associated scope has already been completed normally. This condition is verified by the compensation handler controller before it is able to execute its body activity. Unlike the termination handler, the faults thrown by the activities of the compensation handler are propagated to the scope which has called the faulted compensation handler. This is done by synchronizing on the `Fa` port shared with the scope component interface. Furthermore, in BPEL, the compensation of scopes occurs in the reverse order of their completion. In FIACRE, a simple way to handle such compensation order is to handle it non-deterministically. Otherwise, we have to implement a dynamic mechanism as it is done by [8].

3.3 BPEL Timed Aspects in FIACRE

The modeling of the relative time of BPEL (`For duration`) is straightforward in FIACRE and it could be viewed as a timed event with the same minimal and maximal bounds. Concerning the absolute time of BPEL (namely the Until deadline), it could be handled by explicitly fixing the starting time of the process. In this way, we can easily bound the interval associated with the timed event to : [deadline - start , deadline - start]. Still, this is hardly a practical solution

-especially in the context of verification- since one could argue that the starting date is not always known. In fact, we are interested in verifying the correctness of the system for any starting date.

Modeling of the Absolute Time: The idea is to add to the model a transition which makes it possible for a non-deterministic period of time to elapse between a reference date and the start of the system. This is done [12,20] easily in formalisms based on timed automata. A clock `H` measuring the absolute time is introduced. An initial transition reinitializes all the clocks of the system except `H`. Moreover, this clock is tested in guards ($H \geq D$) in order to know if the absolute time D has been reached. In FIACRE, we do not have explicit clocks, which makes this situation harder to handle. In order to model the absolute time in FIACRE, the two following problems should be considered :

1. Model a non deterministic elapse of time initially: a synchronization on a timed port is introduced.
2. Test whether the absolute time has been reached. This is done by introducing a synchronization on the port `after` being managed by a timer process. Once the deadline is reached, this synchronization becomes non-blocking

The system is built by composing the timer with the component modeling the root activity of the BPEL process:

```
component BPEL_Process is
    port after : none
    par
            after → process_act [after,...,]
        || after → timer [after]
end
```

Managing a Unique Absolute Time. The `Timer` process runs concurrently with the real system and measures when the absolute deadline is reached. We will start by treating the case of a system having one absolute time. The behavior of the process is as given in Fig. 7 :

In order to model the non deterministic delay, we identify two cases in the timer process depending on whether the absolute date has been reached or not. After choosing one of these cases, the process will either wait until the time is reached in the former case or enable an immediate synchronization on the `after` port in the latter case.

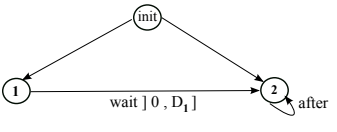


Fig. 7. Absolute time treatment

Managing Multiple Absolute Time. Now that the idea is clear, we will generalize the timer process (Fig. 8) so that we can take into account the case when several absolute times are used in the BPEL description.

In Fig.8, all the absolute times of the system are sorted chronologically. Initially and non deterministically the process is set up to one of the time intervals (i.e strictly before the first, exactly or after the first...). Afterwards, based on what interval has been reached, the process enables a synchronization on the $\texttt{after}_i$ ports associated to all the elapsed dates.

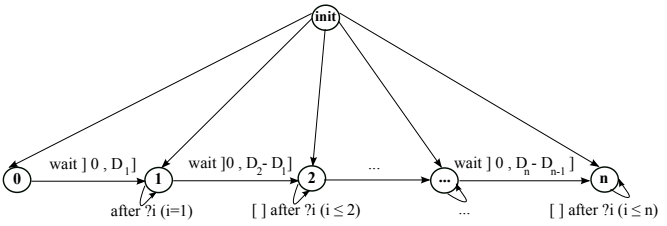


Fig. 8. Absolute time treatment

Other Time Related Constructs. It is reasonable to assume that the elapsed time that represents the waiting for the reception of requests is unbounded. Accordingly, we associate a time interval of $[0, \infty[$ with the synchronization events that model a reception from the environment namely the `receive` activity and the `<on message/>` events of both the pick activity and the event handlers (`port: I_rec`). Moreover, for verification purposes, it is important to consider the reception time of responses resulting from an invocation of web services (`synchronous invoke`) as bounded. This delay represents the duration of treatment of the request by the partner web service. To do so, the BPEL code should be annotated with time constraints attributes [12]. These constraints represent the execution duration of the synchronous invoke activities. The modeling in FIACRE of such time constraints consists of a timed port associated with a time interval having the same bounds. Apart from these constructs, all of the other events are considered as instantaneous (Fig. 9).

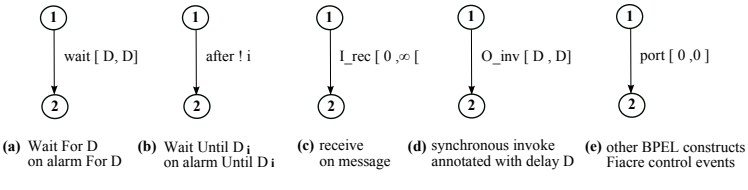


Fig. 9. Summary of timed constructs in FIACRE

4 Verification

The transformation patterns are used for verification purposes. Our verification framework supports three kinds of properties.

1. Structural properties : they express the "well-formedness" of the BPEL code. For instance, we are able to verify that a receive activity is always matched with a reply activity, or that two concurrent receive activities cannot wait

for the same operation. This is done by verifying statically that the faults resulting from such situations are not thrown.
2. Temporal properties : these are the typical properties written in LTL such as the safety, liveness and response properties.
3. Timed properties : they could be written in MITL [2] which is a timed variant of LTL and are handled here using timed observers. These observers may be incorporated either at the FIACRE level or at the BPEL level.

Properties are verified using the Tina toolbox [5] which serves as a model checker for temporal properties on FIACRE models.

Bounded Response Property. The bounded response property could be written as $\Box(receive \Rightarrow \Diamond_{\leq T} finish)$ which means that `finish` must occur within T time units after the end of the receive activity. We note that the specification of complex logical properties could be eased by defining a property pattern language as done in [6,13]. In order to verify such a property on the BPEL process, we need to measure the delay between the reception of a request and its corresponding reply. This verification is done on two levels :

1. Every `receive` activity is followed by a `reply` activity. It is the same as verifying a structural property.
2. The elapse of time between the receive activity and the end of the process is bounded by a fixed time T. This is achieved by defining a timed observer at the BPEL code level (Fig. 10).

The observed system is encapsulated inside a flow where another scope that plays the role of the observer is added. This scope contains an event handler in which the duration of the constraint is fixed. Whenever this duration is reached, a variable `err` is valuated to True reflecting the violation of the property. Furthermore, the event handler should start at the moment of the reception of the first request. This is represented by adding a link (`lnk1`) from the receive activity to the scope of the event handler. Moreover, at the end of the observed activity, the event handler should be terminated. That is why another link (`lnk2`) is added from the observed activity to the primary activity of the scope (empty).

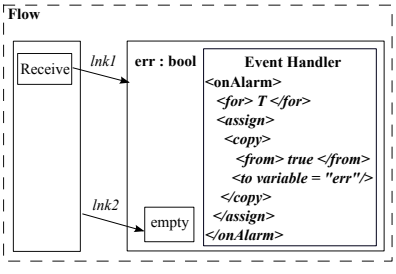


Fig. 10. Timed observer

5 Case Study

We have applied our transformation on the purchase order example taken from the BPEL specification release [3]. We have made some modifications in order to show how the absolute time is treated in our technique. After the modifications, the service is only offered until a deadline is reached (Fig. 11).

In fact, upon the reception of the purchase request, the process initiates simultaneously two sequences of activities which are used for calculating the final price and finalizing the shipment. Once the two tasks are completed, the process replies an invoice to the customer. However, a deadline inside of an event handler is fixed. After this deadline is reached, the event handler will proceed by replying a negative response to the customer and by terminating the whole process using an exit activity. Likewise, whenever a fault is thrown during the execution of the purchase sequence, the fault handler will terminate the process and reply a fault to the customer.

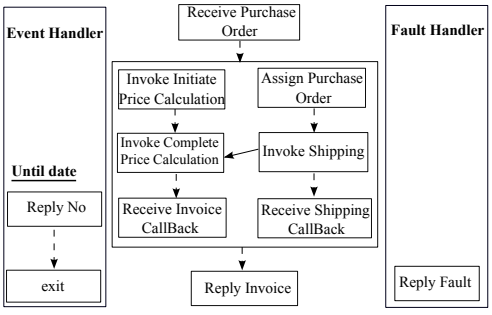


Fig. 11. Timed purchase example

Use Case Modeling and Verification. We give in the following the root component of the FIACRE code associated to the case study. In this excerpt, only the useful ports and shared variables are shown.

```
component BPEL_process [I_rec: inputs, O_rep: outputs,
                        O_inv: inputs, I_inv: outputs] is
port S,F,after,stpd: none,
     faIN,fa: fault
var stop: bool= false,
    link: t_link,
    vars: t_var
par /* BPEL process pattern with timer */
    after,faIN,stpd,exit →
      process_act [S,F,I_rec,O_rep,I_ink,O_ink,faIN,stpd,
                   exit,after](& link,&stop,&var)
 || faIN,stpd,exit →
      fault_handler [faIN,I_rec,O_rep,O_ink,I_ink,
                     fa,stpd,exit](&link, &stop, &var)
 || after → timer [after]
end
```

On one hand, we compose the primary activity of the process which contains the `purchase` sequence and the event handler and on the other the fault handler. Additionally, the `Timer` process runs concurrently and synchronizes on the `after` port with the system. Since the event handler contains an `exit` activity, a synchronization on the port `exit` with the fault handler is permitted. This triggers a termination mechanism done by communications through the shared variable `stop` and the port `stpd`. Also, the `process_act` may generate faults by synchronizing with the fault handler component on the `faIN` port. This leads to the termination of the `process_act` component.

Now, properties may be verified on the underlying code. As an example, we show the following *response* property which guarantees that independently of the starting date of the system, a purchase demand is always followed by a reply.

$$\Box(\ purchase\_demand \Rightarrow \Diamond(Reply\_Invoice \vee Reply\_Fault \vee Reply\_No))$$

6 Related Works

Various works have been pursued in the area of modeling and verification of BPEL processes. Major part of the literature have addressed a mapping from BPEL to process algebra languages as CCS [16] and LOTOS. Indeed, in [21] the authors show how CCS can be exploited to treat BPEL constructs used to verify CTL properties. Moreover, a two-way mapping between BPEL and LOTOS is studied in [8]. This technique is quite interesting because it allows the conception of web services at the two levels. It also allows to reason about the equivalences between services by making use of the bisumilation notion. The second group of work was interested in mapping BPEL constructs to Petri Nets. The first to address an extensive mapping to Petri Nets was the work of [10] in which one could verify properties like the absence of deadlock. The work of [10] has been extended in [14] in order to cover the new features of BPEL 2.0.

Others have been interested in transformations towards Promela. In the work of [18], a subset of the BPEL constructs are transformed into Promela and connected to other processes in the description. LTL properties are then verified on the result process by applying the Spin model checker [11].

Approaches based on proof methods have also been used. In [1] a mapping from BPEL to Event_B has been done. This technique does not face the problem of state number explosion. However, the proof obligations produced by the Event_B require an interactive assistance and are usually complex to achieve. Compared to them, these works cover only the untimed aspects of BPEL. That is, the time is abstracted and the choice is resolved in a non-deterministic way.

Finally, some works were interested in modeling the timed aspects of BPEL. We quote in this matter [12] in which the timed behavior of BPEL activities are captured by introducing a formalism (WSTTS) based on timed automata. In this work, the absolute time of BPEL is treated. They support as well the expression of complex timed properties in Duration Calculus and verify them using NuSmv. The drawback of their technique is that they only handle a discrete notion of time while we consider a dense model of time. Moreover, our treatment of the absolute time in FIACRE is based on a formalism based on Timed Petri nets with priorities. Another work based on a transformation towards timed automata (TA) is also presented in [20]. An algorithm for mapping these patterns to timed automata is later integrated [19] in the `ActiveBPEL` tool. This work covers both the relative and the absolute time of BPEL. However, the compensation and the termination handlers are not supported. Moreover, their transformation does not provide a way to explicitly specify the duration of synchronous calls or complex time-related properties. In [15], a transformation from BPEL to discrete time LTS is studied. Again, in our technique, we consider a dense model of time. In [22], the authors are interested in reasoning about the duration between two activities by modeling the timed behavior of BPEL in timed Petri Nets. Nevertheless, the absolute time is treated by assuming that the current time is given explicitly. This is not a solution because a unique verification cannot guarantee the correctness of the system independently of its starting time.

Project	[21,8]	[10,14]	[18]	[1]	[12]	[20]	*This*
Formalisms	LOTOS/CCS	Petri Nets	Spin	Event B	WSTTS	TA	FIACRE
BPEL 2.0	–	+	–	+	+	–	+
Coverage	+	++	+	++	+	+	++
Time Modeling	–	–	–	–	+	+	+
Timed Properties	–	–	–	–	+	–	+

Fig. 12. Brief BPEL verification tools coverage

7 Conclusion and Future Works

We have presented a transformation from WS-BPEL 2.0 to the FIACRE specification language. The transformation is mostly structural and is based on a set of patterns covering both behavioral and timed aspects of BPEL. We have also shown how timed properties can be expressed in our framework. To do so, a novel idea that consists in building an observer at the BPEL level is given. We have experimented the proposed transformations on a case study with absolute date dependent behaviors. Properties can be verified on this model independently from the starting date of the system.

For the continuation of the work, we are currently studying how to incorporate ways in the Fiacre language itself to define the proposed patterns so that they can be instantiated to build the FIACRE encoding of the BPEL model.

Moreover, for the time being, the properties are specified logically in LTL and MITL which sometimes makes it complex to handle. Our current work deals with defining a property pattern language used for the specification of complex timed properties as in [13].

References

1. Ait-Sadoune, I., Ait-Ameur, Y.: A proof based approach for modelling and verifying web services compositions. In: Proceedings of the 2009 14th IEEE International Conference on Engineering of Complex Computer Systems, ICECCS 2009, pp. 1–10. IEEE Computer Society, Washington, DC, USA (2009)
2. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality (1996)
3. Alves, A., Arkin, A., Askary, S., Bloch, B., Curbera, F., Goland, Y., Kartha, N., Sterling, König, D., Mehta, V., Thatte, S., van der Rijn, D., Yendluri, P., Yiu, A.: Web Services Business Process Execution Language Version 2.0. OASIS (May 2006)
4. Berglund, A., Boag, S., Chamberlin, D., Fernández, M.F., Kay, M., Robie, J., Siméon, J.: XML Path Language (XPath) 2.0 (W3C Recommendation) (January 2007)
5. Berthomieu, B., Ribet, P.-O., Vernadat, F.: The tool TINA – construction of abstract state spaces for petri nets and time petri nets. International Journal of Production Research 42 (2004)
6. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: Proceedings of the 21st International Conference on Software Engineering, ICSE 1999, pp. 411–420. ACM, New York (1999)

7. Farail, P., Gaufillet, P., Peres, F., Bodeveix, J.-P., Filali, M., Berthomieu, B., Rodrigo, S., Vernadat, F., Garavel, H., Lang, F.: FIACRE: an intermediate language for model verification in the TOPCASED environment. In: European Congress on Embedded Real-Time Software (ERTS) (2008)
8. Ferrara, A.: Web services: a process algebra approach. In: ICSOC 2004: Proceedings of the 2nd International Conference on Service Oriented Computing, pp. 242–251. ACM Press, New York (2004)
9. Henzinger, T.A., Manna, Z., Pnueli, A.: Timed Transition Systems. In: REX Workshop, pp. 226–251 (1991)
10. Hinz, S., Schmidt, K., Stahl, C.: Transforming BPEL to Petri Nets. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 220–235. Springer, Heidelberg (2005)
11. Holzmann, G.J.: The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley Professional, Reading (2003)
12. Kazhamiakin, R., Pandya, P., Pistore, M.: Representation, verification, and computation of timed properties in Web Service Compositions. In: Proceedings of the IEEE International Conference on Web Services, pp. 497–504. IEEE Computer Society, Washington, DC, USA (2006)
13. Lanz, A., Weber, B., Reichert, M.: Workflow time patterns for process-aware information systems. In: Aalst, W., Mylopoulos, J., Sadeh, N.M., Shaw, M.J., Szyperski, C., Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) A Concurrent Pascal Compiler for Minicomputers. LNBIP, vol. 50, pp. 94–107. Springer, Heidelberg (2010)
14. Lohmann, N.: A feature-complete Petri net semantics for WS-BPEL 2.0. In: van Hee, K., Reisig, W., Wolf, K. (eds.) Proceedings of the Workshop on Formal Approaches to Business Processes and Web Services (FABPWS 2007), pp. 21–35. University of Podlasie (2007)
15. Mateescu, R., Rampacek, S.: Formal Modeling and Discrete-Time Analysis of BPEL Web Services. International Journal of Simulation and Process Modeling (2008)
16. Milner, R.: A Calculus of Communicating Systems. Springer-Verlag New York, Inc., Secaucus (1982)
17. Moreau, J.J., Chinnici, R., Ryman, A., Weerawarana, S.: Web Services Description Language (WSDL) version 2.0 part 1: Core language. Candidate recommendation, W3C (March 2006)
18. Nakajima, S.: Model-Checking Behavioral Specification of BPEL Applications. Electron. Notes Theor. Comput. Sci. 151, 89–105 (2006)
19. Pu, G., Zhao, X., Wang, S., Qiu, Z.: Towards the semantics and verification of BPEL4WS. Electron. Notes Theor. Comput. Sci. 151, 33–52 (2006)
20. Qian, Y., Xu, Y., Wang, Z., Pu, G., Zhu, H., Cai, C.: Tool Support for BPEL Verification in ActiveBPEL Engine. In: 18th Australian Software Engineering Conference ASWEC 2007, pp. 90–100 (April 2007)
21. Salaun, G., Bordeaux, L., Schaerf, M.: Describing and reasoning on web services using process algebra. In: IEEE International Conference on Web Services, p. 43 (2004)
22. Song, W., Ma, X., Ye, C., Dou, W., Lü, J.: Timed modeling and verification of BPEL processes using time petri nets. In: Proceedings of the 2009 Ninth International Conference on Quality Software, QSIC 2009, pp. 92–97. IEEE Computer Society, Washington, DC, USA (2009)

A Maturity Model Assessing Interoperability Potential

Wided Guédria[1,2], Yannick Naudet[2], and David Chen[1]

[1] IMS-LAPS, University Bordeaux1. 351, Cours de la Libération,
33405, Talence Cedex, France
[2] CITI, Henri Tudor Public Research Center. 29, Av. John F.Kennedy,
1855 Luxemboug-Kirchberg, Luxembourg
{wided.guedria,Yannick.naudet}@tudor.lu

Abstract. In a globalized and networked society, interoperability is a pervasive topic as it is a key factor of success for enterprises to meet their own added values and to exploit the market opportunities. This paper aims at presenting a model based on enterprise interoperability potential following the maturity models approach. Interoperability potential assessment requires a framework to capture the artifacts needed for enterprise interoperations. The framework of Enterprise Interoperability (FEI), currently under CEN/ISO standardization process is used as basis for the defined model.

Keywords: Enterprise Interoperability, framework, interoperability potential, maturity model, assessment.

1 Introduction

In the past, most companies created their own applications and designed their own set of services, but times have changed. In the current globalized and networked society, enterprises need to collaborate with other enterprises to meet their own added values and to exploit the market opportunities. A major issue in global collaboration and co-operation is the development of interoperability. Interoperability is the *Ability of two or more systems or components to exchange information and to use the information that has been exchanged* (IEEE) [1].

In order to support enterprises to better interoperate with their partners, clients, providers, etc; enterprise interoperability requires being assessed and continuously improved. One of the assessment methods is concerned with the use of maturity models. A maturity model is a framework that describes, for a specific area of interest, a number of levels of sophistication at which activities in this area can be carried out [2]. In our case, the specific area of interest is Enterprise Interoperability. Enterprise interoperability maturity can be measured in two ways : *A priori* where the measure relates to the potentiality of a system to be interoperable with a possible future partner whose identity is not known at the moment of evaluation, *A posteriori* where the measure relates to the compatibility measure between two (or more) known systems willing to interoperate.

The most known interoperability maturity models such as: LISI (Levels of Information System Interoperability) [3], OIM (Organizational Interoperability Model) [4],

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 276–283, 2011.

LCIM (Levels of Conceptual Interoperability Model) [5], and EIMM (Enterprise Interoperability Maturity Model) [6], deal with the *a posteriori* measure of interoperability and do not sufficiently address potentiality of interoperability. Moreover, they focus on one single facet of interoperability (data, technology, conceptual, Enterprise modeling, etc.). Some comparison study has been reported in [7] and [8].

The objective of this paper is to focus on *a priori* measure of enterprise interoperability potential. The Framework for Enterprise Interoperability (FEI) initially elaborated in INTEROP NoE [10] and now under CEN/ISO standardization process (CEN/ISO 11354) is used as a basis to build this maturity model.

The paper is structured as follows. In section 2 we present the detailed specification of the maturity model for enterprise interoperability. An industrial application example is outlined in section 3. Finally section 4 concludes the paper and proposes future work.

2 The Maturity Model for Enterprise Interoperability Potential

In this section, MMEI is detailed. It covers the whole problem space of the Framework for Enterprise Interoperability [11]: four enterprise concerns (data, service, process, business) and three aspects of interoperability (conceptual, technological and organizational).

2.1 The Scope of MMEI

MMEI is intended to be used by people who are concerned by the assessment of enterprise interoperability and by the detection of which might need to be improved to meet the needs and ambitions of the enterprise.

For that, we need to collect information through a series of interviews. The content of the assessment interview depends on the assessment scope and the enterprise needs. From the interviews, a rating shall be assigned based on validated data. Conclusions are taken by the assessor team after analysis.

MMEI as defined in [9] needs efforts from evaluators to perform an assessment and especially to prepare the evaluation sheets for interviews. This is due to the combination of *a priori* and *a posteriori* measurements of interoperability. Considering that the objective of the assessment should be specified at the beginning, the MMEI as defined in [9] cannot be used directly without any transformation. It has to be specialized in either *a priori* or *a posteriori* measurement in order to be easily applicable. The next section details the MMEI model, assessing the enterprise interoperability potential.

2.2 Overview of MMEI

MMEI allows determining in early stages whether meaningful interoperability between enterprises is possible. It defines five levels of interoperability maturity as presented in table 1.

Table 1. Overview of MMEI levels

Maturity Level	*Maturity capability*
Level 4 – Adaptive	Capable of negotiating and dynamically accommodating with any heterogeneous partner
Level 3 – Organized	Capable of meta modeling to achieve the mappings needed to interoperate with multiple heterogeneous partners
Level 2 – Aligned	Capable of making necessary changes to align to common formats or standards
Level 1 – Defined	Capability of properly modeling and describing systems to prepare interoperability
Level 0 – Unprepared	Not relevant: there is no capability for interoperation

2.3 Specification of MMEI

Each maturity level is described by a matrix following a simplified version of the FEI containing only two basic dimensions: "interoperability level" and "enterprise concern". For each combination of these dimensions, for all levels of maturity, tasks and activities are specified. They give an indication about the state of the enterprise (the current level of the enterprise) or show where investigations are required to achieve desired maturity levels.

Level 0 (Unprepared). At this level, systems run stand-alone and are not prepared for interoperability. Possible Communication with external systems remains mainly manual exchange. There is no willingness to work in an open and collaborative environment. As this level is characterized by proprietary or closed systems, table 2 presents a description of existing problems that have to be solved in order to prepare interoperability.

Table 2. Description of the MMEI level 0

	Conceptual	Technological	Organizational
Business	Visions, strategies, politics not properly formulated or documented.	No IT infrastructure or platform in place.	Responsibilities/ authorities not defined.
Process	Processes not formally described.	Manual processes.	(Idem.)
Service	Services not formally defined.	Stand-alone services.	(Idem.)
Data	Data representation, not completely modeled.	Closed data storage devices, manual exchange.	(Idem.)

Level 1 (Defined). Starting from this level, the system is considered open to interoperability. This means there is a will to prepare the system for future interoperations. The description of level 1 is shown in the table 3.

Table 3. Description of the MMEI level 1

	Conceptual	Technological	Organizational
Business	Models are defined and documented.	IT infrastructure / platform in place, and connectable.	Responsibilities / authorities defined and in place.
Process	(Idem.)	Platform dependant Processes.	(Idem.)
Service	(Idem.)	Platform dependant Services.	(Idem.)
Data	(Idem.)	Connectable devices or simple electronic exchange possible.	(Idem.)

Table 4. Description of the MMEI level 2

	Conceptual	Technological	Organizational
Business	Use of standards to facilitate alignment with other models.	Use of standards to facilitate the alignment with other IT infrastructures / platforms.	Organization structure can be adjusted.
Process	(Idem.)	Process modeling and execution tools using standards.	Procedures of work defined and adjustable.
Service	(Idem.)	Use of standards to facilitate the alignment with other Service execution platforms.	Guidelines for service exchanges in place and can be adjusted.
Data	(Idem.)	Databases are connectable, remote access to databases possible.	Rules and methods for data interoperability management in place and can be adjusted.

Level 2 (Aligned). This level of maturity requires that the system has the ability (i.e. has the capabilities) to make necessary changes in its components in order to adhere to standards. The description of level 2 is shown in the table 4.

Level 3 (Organized). At this level, Organization and decision-making are decentralized to improve flexibility and reactivity. The use of ontology, reference or standardized meta-models is required in order to be able to have future interoperation with different and heterogeneous partners. Level 3 requires that people have been trained with collaborative approaches and interoperability methods and guidelines. The description of level 3 is shown in the table 5.

Table 5. Description of the MMEI level 3

	Conceptual	Technological	Organizational
Business	Business models designed for multi partnership and networked enterprise.	Open infrastructure/ platform.	Organization is trained for interoperability.
Process	Process specification for mapping based on meta-models.	Platforms allowing collaborative execution of processes.	Guidelines for cross-enterprise collaborative processes.
Service	Services annotation based on meta-models.	Composable services.	Multiple roles and responsibilities allowing more flexible service management.
Data	Meta models defined for possible future mappings.	Remote access to databases possible for applications.	Flexible rules and methods for data interoperability management.

Table 6. Description of the MMEI level 4

	Conceptual	Technological	Organizational
Business	Continuous Business and IT alignment.	Reconfigurable IT infrastructure and platforms.	Agile organization for on-demand business.
Process	Dynamic process re-engineering.	Platform-independent dynamic and adaptive tools and engines for processes.	Real-time monitoring of processes, adaptive work procedures.
Service	On-demand and adaptive service modeling.	Platform-independent and reconfigurable services architecture.	Dynamic and on-demand allocation of resources to services.
Data	Adaptive data models (both syntax and semantics).	Direct database exchanges capability and full data conversion tool.	Adaptive data management rules and methods.

Level 4 (Adaptive). This is the highest level where interoperability itself becomes a subject of continuous improvement (evolution and adaptation). Level 4 is rarely reached by most of enterprise systems. Description of level 2 is shown in the table 6.

2.4 Coverage of Existing Maturity Models

In this paper, MMEI deals with the *a priori* measurement of interoperability which is not addressed by the existing maturity models. This makes it different from others. However, MMEI covers all concerns of an enterprise in terms of interoperability and can be also exploited for *a posteriori* assessment with slight adaptations. In this perspective, the Fig.1 from [8] shows how it covers existing maturity models for interoperability.

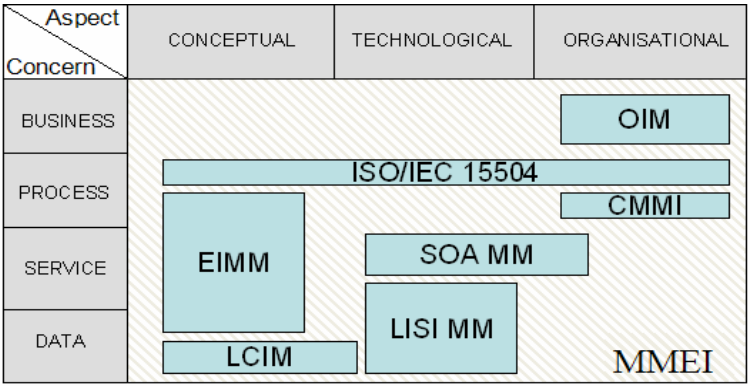


Fig. 1. Coverage of MMEI dimensions by existing maturity models

3 Case Study

To illustrate and validate the proposed maturity model, we present here a case study of a multinational company. METS (Manufacture Electro-Technical of Sousse) is part of the German group Dräxlmaier [12], which is specialized in automobile manufactures with modern wiring harness systems, exclusive interiors and electrical components. In order to meet high standard, the company has established a management system which aims to continuously optimize their processes and therefore increasing its competitiveness: the Dräxlmaier Process Management (DPM). The quality of this management system is regularly certified by TS 16949 [12].

The business processes of the company were determined under the consideration of the principle "as much uniformity as possible, as much individuality as necessary" and the existing platforms were conceived under this principle. This leads to synergy effects due to standardization, while also allowing enough flexibility for the integration of varying external requirements.

The process model serves as a communication platform. It contains the process structure and therefore supports the exchange of information within and between networks. Moreover, a process cockpit exists in order to evaluate process performance using key performance indicators (KPI). The working methods are called "work instructions". These instructions are formalized using standards, validated by the quality department, known by all employees, applied and categorized by department. Each employee has a functional file where we find his role with the detailed activities to be performed (role description) and the persons that are able to replace him in each activity in case of absence.

In order to ensure a regular monitoring of the site, the headquarters, in Germany, require that a "Portfolio Management" is weekly sent. It is a document containing all ongoing tasks, problems faced, next to do...

We present here the assessment of the process interoperability concern to evaluate its interoperability potential. Based on the collected information (through a series of interviews), we have completed an evaluation sheet (see table 7).

Table 7. Description of the MMEI level 3.

Activities to evaluate	Observations	Team Rating			
		NA	PA	A	FA
Use of standards to facilitate alignment with other models.	Use of the DPM standard.	☐	☐	☐	☒
Process modeling and execution tools using standards.	The standardized process model serves as a communication platform.	☐	☐	☐	☒
Procedures of work defined and adjustable	Instructions are flexible.	☐	☐	☐	☒
Process specification for mapping based on meta-models.	----------	☒	☐	☐	☐
Platforms allowing collaborative execution of processes.	Platforms were conceived under the collaboration principle.	☐	☐	☐	☒
Guidelines for cross-enterprise collaborative processes.	Process model contains instructions to exchange information between the process networks.	☐	☐	☐	☒
Dynamic process re-engineering.	----------	☒	☐	☐	☐
Platform-independent and adaptive tools and engines for processes.	----------	☒	☐	☐	☐
Real-time monitoring of processes, adaptive work procedures.	Regular monitoring (weekly)	☐	☒	☐	☐

The ratings are given by the assessors based on the achievement degrees of the activities being evaluated. Clearly, it is difficult for people to make such fine judgment, especially in our case where the achievement degree is not a binary one but a graduated state. We won't detail the used metrics here. However a specificity of our approach is that behind this evaluation, we have used the linguistic variables to facilitate the task of the assessors to find suitable scores according to their observations upon the enterprise. We have defined the linguistic variable [9] "state of an activity" as rating the following values: Not achieved (NA), Partially Achieved (PA), Achieved (A) and Fully Achieved (FA). Each assessor chooses a value among latter ones to qualify the practices achievements. From these linguistic values, scores are assigned, based on previously defined membership functions [13]. Each one of the team ratings, presented in the evaluation sheet (cf. table 7), is calculated by aggregating the assessors' scores using the OWA operator [14]. Finally, we use fuzzy rules [13] to find the reached maturity level. According to team ratings of this use case (cf. table 7), the reached level regarding interoperability potential is 3. Instructions are then given to fill requirements towards the next level (level 4).

3 Conclusion

The assessment is an activity that can be performed either as part of an improvement initiative or as part of a maturity determination approach. The first step to be done in an assessment process is to define its purpose (why it is being carried out), its scope,

what constraints apply to the assessment and any additional information that needs to be gathered. In this paper, we have proposed a maturity model for enterprise interoperability (MMEI). Five levels of maturity were defined and described. Based on the FEI, MMEI covers the four main enterprise interoperability concerns and the three main interoperability problems which were usually dealt by separated distinct maturity models. Future work is planned to refine the proposed model and metrics, and to perform some more detailed case studies in enterprises.

References

1. IEEE (Institute of Electrical and Electronics Engineers) standard computer dictionary, a compilation of IEEE standard computer glossaries (1990)
2. Alonso, J., De Soria, I., Orue-Echevarria, L., Vergara, M.: Enterprise Collaboration Maturity Model (ECMM): Preliminary Definition and Future Challenges. In: Enterprise Interoperability IV, Part VII, pp. 429–438. Springer, Heidelberg (2010)
3. C4ISR Interoperability Working Group, Levels of information systems interoperability (LISI), Tech. report, US Department of Defense, Washington (1998)
4. Clark, T., Jones, R.: Organizational interoperability maturity model for c2. In: Proc. of the 1999 Command and Control Research and Technology Symposium, Washington (1999)
5. Tolk, A., Muguira, J.A.: The levels of conceptual interoperability model. In: 2003 Fall Simulation Interoperability Workshop, USA (2003)
6. ATHENA.: Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Applications, FP6-2002-IST1, Integrated Project Proposal (2003)
7. Panetto, H.: Towards a classification framework for interoperability of enterprise applications. International Journal of Computer Integrated Manufacturing 20(8), 727–740 (2007)
8. Guédria, W., Naudet, Y., Chen, D.: Interoperability Maturity Models, survey and comparison. In 3rd IFAC/IFIP, OTM Workshop, Monterrey, pp. 273–282 (2008)
9. Guédria, W., Naudet, Y., Chen, D.: A Maturity Model for Enterprise Interoperability. In: 4th IFAC/IFIP, OTM Workshop, Vilamoura, pp. 216–225 (2009)
10. INTEROP, Enterprise Interoperability -Framework and knowledge corpus- Final report, INTEROP NOE, Contact n 508011, Deliverable DI.3 (2007)
11. CEN/ISO 11354-1, Part 1: Framework for enterprise interoperability, Draft International Standard ISO/DIS 11354-1 (2009)
12. Official website of the Dräxlmaier Group, `http://www.draexlmaier.de`
13. Zadeh, L.A. Soft computing and Fuzzy logic. IEEE Software (1994)
14. Yager, R.R.: Aggregation operators and fuzzy systems modeling. Fuzzy Sets and Systems 67, 129–145 (1994)

Semantic Support for Security-Annotated Business Process Models

Ioana Ciuciu[1], Gang Zhao[2], Jutta Mülle[3], Silvia von Stackelberg[3], Cristian Vasquez[1], Thorsten Haberecht[3], Robert Meersman[1], and Klemens Böhm[3]

[1] STARLab, Vrije Universiteit Brussel, Brussels, Belgium
[2] Intelartes SPRL, Waterloo, Belgium
[3] Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany
{iciuciu,cvasquez,meersman}@vub.ac.be,
{gang.zhao}@intelartes.com,
{jutta.muelle,silvia.stackelberg,
thorsten.haberecht,klemens.boehm}@kit.edu

Abstract. Service-Oriented Architectures (SOA) benefit from business processes (BP), which orchestrate web services (WS) and human actors in cross organizational environments. In this setting, handling the security and privacy issues while exchanging and processing personal data is essential. This lacks for secure business processes management. To achieve this, we represent security constraints descriptively by annotating process models, aiming to enforce these constraints by a secure business process management system (BPMS).To assist the process modeler in annotating process models, we introduce in this paper a tool which provides semantic interoperability during process design. By enforcing a shared conceptualization (ontology) of the security and privacy domains with an ontology base grounded in natural language this tool called knowledge annotator is able to make annotation recommendations according to knowledge stored in a knowledge base. The annotator is validated in an employability use case scenario.

Keywords: Business Process Model, Semantic Annotation, Semantic Interoperability, Ontology, Knowledge Management, Security Constraints, Employability.

1 Introduction

The Trusted Architecture for Securely Shared Services (TAS^3)[1] project provides a next generation trust and security architecture that is ready to (1) meet the requirements of complex and highly versatile business processes, (2) enable the dynamic, user-centric management of policies and (3) ensure secure end-to-end transmission of personal data and user-controlled attributes between heterogeneous, loosely coupled, context-dependent systems.

[1] http://tas3.eu/

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 284–298, 2011.

One of the challenges, in this context, is to offer a secure business processes framework for sharing, accessing, and using personal data processing services in federated environments.

To make business processes secure, we proceed as follows. We annotate the business process model with security constraints in the first step. These annotations concern the handling of authentication, authorization, audit logging, and other security issues. The business process management system (BPMS) transforms security annotations into descriptive security policies or triggers process model extensions. Finally it executes secure business processes by dedicated system components. For example, these components allocate actors to activities, enforce data-specific authorizations, or trigger security-specific user involvements. This infrastructure guarantees that business processes will be performed according to the annotated security constraints. In order to ensure semantic interoperability between the different components of the system, we provide an ontology, embedded in the architecture, which explicitly documents the relationship between core security concepts. One goal is that all security-relevant business process specifications are annotated to a common, agreed upon semantic knowledge structure (ontology) in order to ensure alignment and interoperability between actors with respect to security concepts.

It is therefore of major importance to have a mechanism which ensures the correct specification of the security annotations. The solution we propose in this paper is a semantic security annotation tool for business processes. This semantic annotator tool aims to assist the process modeler in specifying security constraints for business process models. It uses a lower common ontology representing security constraints for business processes and a knowledge base storing a set of previously defined correct annotated rules. The system is able not only to support the process modeler with syntactically correct security concepts, but also to assist him with annotation suggestions. The suggestions are made according to information retrieved from the knowledge base which is matched against the process modeler input (knowledge).

The rest of the paper is organized as follows: Section II briefly describes related work. Section III provides background information on the technology being used. The approach is presented in Section IV. Section V shows the possible annotation use cases within an employability use case scenario developed by the University of Nottingham [1]. Section VI presents our conclusion and suggestions for future work.

2 Related Work

The idea of adding semantics to business processes has been adopted and its importance has been recognized by the business process community for several years now [2,3]. Ever since, Semantic Web technologies have been applied and new tools have been proposed in order to add semantics to business processes. The semantics are captured via semantic annotations specifying the process dynamics and behavior [4], or the meaning of process elements (as in e.g. the SUPER[2] project [5]).

Several semantic annotation models have been proposed, aiming at semantic interoperability of business process knowledge [6]. The focus of this research is on annotation tools aiming at assisting the process modeler with an ontology-based

[2] http://www.ip-super.org/

recommendation system. Betz [7] proposes an approach for the automatic user support based on an autocompletion mechanism during the modeling process (where business processes are represented as Petri Nets). Born [8] presents a tool for the user-friendly integration of domain ontology information in the process modeling, through match matching and filtering techniques. A similar approach, based on linguistic analysis of process element labels and of the concept names is presented in [9] in order to support process modelers with annotation suggestions.

Our approach is grounded in natural language. It is built on the ontology-based data matching principles [10]. The goal is to assist the process modeler with annotation suggestions retrieved from a security constraints ontology base and from a knowledge base storing previously defined annotations.

3 Background

We represent business process models as Business Process Model and Notation 2.0[3] (BPMN 2.0) diagrams. BPMN is the widely accepted de-facto standard for process models(OMG[4], 2011). It provides several elements to represent a process flow, such as pools and lanes, activities, events, data objects, flow objects, and artifacts. As BPMN artifacts enable the annotation of process diagrams, we embed security constraints as security-marked annotations into BPMN diagrams. Consequently, the security annotations are standard-conform to BPMN.

The knowledge annotator presented in this paper is based on Developing Ontology Grounded Methodology and Applications (DOGMA, [11]). DOGMA is a formal ontology engineering framework applying the principles of database design methodology (NIAM/ORM2, [12]) to ontology engineering. DOGMA ontology is grounded in natural language and based on the *double articulation principle* [13], which makes the ontology two layered:

1. The *lexon* base layer, containing a set of simple binary facts, called lexons;
2. The *commitment* layer that formally defines rules and constraints by which applications may make use of the lexons from the lexon base.

A lexon is defined as a quintuple $\langle \gamma, t_1, r_1, r_2, t_2 \rangle$ representing a fact type. γ is a context identifier that points to a context where two terms, t_1, t_2 are originally defined and disambiguated. r_1, r_2 are two roles that characterize the relationship between t_1 and t_2. For example, *‹SecBP, Security Annotation, defined for, annotated with, BPMN Element›* is a lexon which means “in the context of secure BP (SecBP), a security annotation is defined for a BPMN element and a BPMN element is annotated with a security annotation”. This example is depicted in Fig. 1.

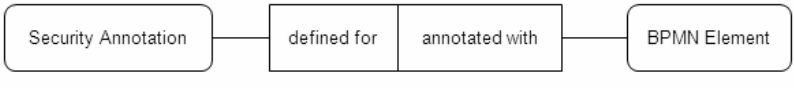


Fig. 1. A lexon example

[3] http://www.bpmn.org/
[4] http://www.omg.org/

A commitment contains a constraint on a (set of) lexon(s). For instance, we can apply the mandatory constraint on the above lexon, "there exists at least one BPMN element per annotation type". The commitment language needs to be specified in a language such as OWL[5] or SDRule language [14].

4 Knowledge Annotator for Secure Business Process Models

The knowledge annotator is designed as a user-friendly, intelligent system, intended to assist the process modeler during the specification of the security-specific constraints and to learn from the process modeler by using a dedicated knowledge base. This is realized by capturing the process modelers' modeling intentions via a user-friendly interface (UI) and by presenting him/her with recommendations. The recommendations are determined before by an ontology-based data matching operation between the user input, the security constraints ontology (see Section 4.2), and the collected security annotations retrieved from the knowledge base (see Section 4.4).

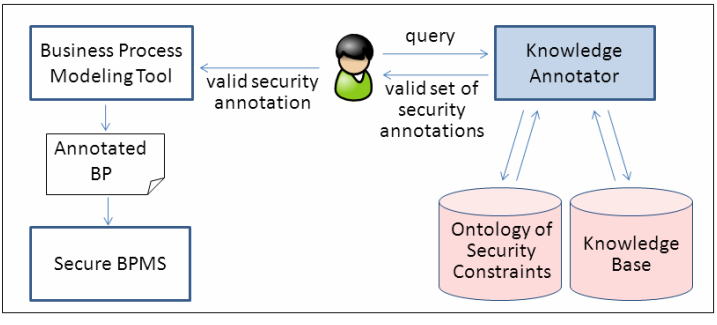


Fig. 2. User-system interactions for the annotation of security constraints

In our approach, the business process model is annotated with security constraints that make use of the concepts from the ontology of security constraints and of the knowledge stored in a knowledge base, as shown in Fig. 2.

Let us take the particular case when the process modeler needs to annotate an "activity" BPMN element using a "requestToUser" type of security annotation. For this he/she sends a query to the annotator, indicating "request to user" in the annotation search fields. The BPMN element of type "activity" is inferred from the business process (BP) modeling tool and passed to the query as well. The annotator assists the process modeler by performing several operations, as shown in Fig.3.

The embedding of the annotator tool into the design phase of security-annotated business processes eliminates the tedious task of manual search for the different options of correct annotations for a specific user query. It presents the modeler with a complete set of options according to the expressed modeling intentions (queries). This is extremely helpful in case of large sets of security annotations stored in the knowledge base and retrieved by the annotator components (see Section 4.5).

[5] http://www.w3.org/TR/owl-ref/

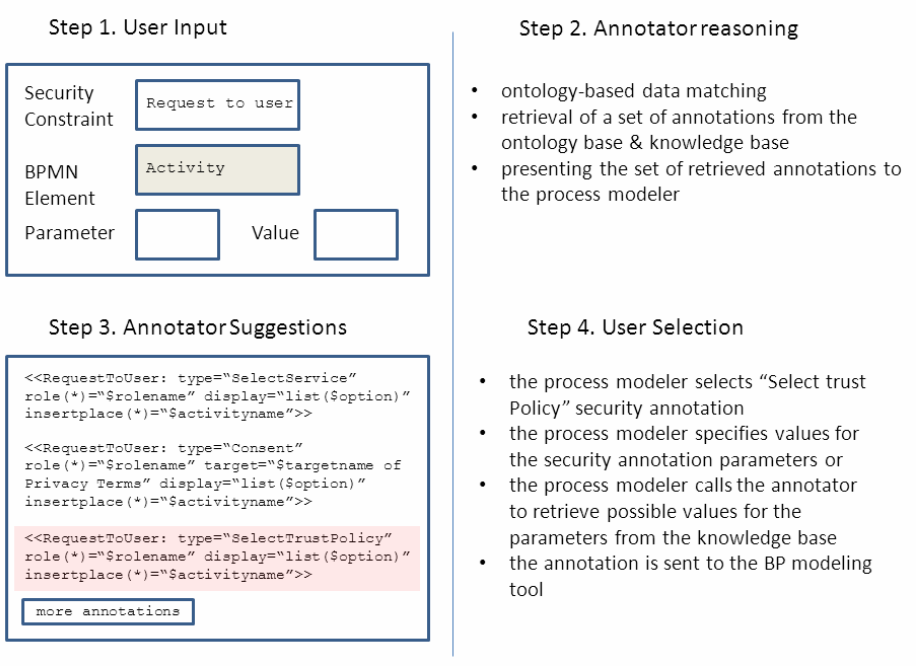


Fig. 3. Example of user-system interactions for a specific security annotation ("RequestToUser")

The ontology-based data matching operation ensures the exploration of different security ontologies developed in the past and the mapping of concepts between them and the ontology of security constraints (see Section 4.3). This allows a wide set of search options for the process modeler when he/she performs the search.

4.1 Security Annotation

A security annotation is a text annotation attached to a BPMN element. The syntax of a security annotation is specified by an annotation term, followed by a list of parameters (mandatory or optional) with their corresponding values:

```
<<AnnotationTerm: list(parameter="value")>>.
```

Our security language supports auditing, authentication, authorization, data and messageflow security, delegation, and user interactions [15].

Fig. 4 gives an example of several annotations of two BPMN lanes ("Placement Advisor" and "Student"). The <<Authn … >> annotations enforce authentications for all process participants executing tasks of these lanes according to the specified parameters, namely authentication "attributes" and the identity provider ("idp"). The role assignments of lane "Placement Advisor" and lane "Student" (<<Assignment type="Role">>) mean that only particular role holders, namely "Placement Advisors" or "Students" have the authorizations to execute tasks. The annotation <<BoD>> for

lane “Student” describes that all tasks must be performed by the same person (binding of duty), while the execution of tasks of lane “Placement Advisor” can be delegated due to spec=”weak” to other role holders.

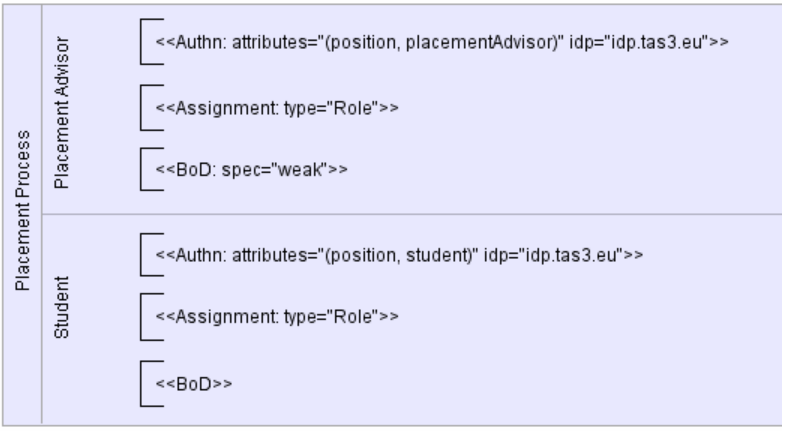


Fig. 4. Security annotations for BPMN lanes

4.2 Ontology of Security Constraints

A lower common ontology has been created to represent the security constraints applying to business processes. The security constraints ontology is used to assist the process modeler (see the approach described in [16]) for annotating the following BPMN 2.0 elements: activities, groups of activities, pools and lanes, data, message flows, and events.

The taxonomy of BPMN elements that can be annotated is illustrated in Fig. 5. Activities are considered to be either tasks or sub-processes. Data subsumes, according to the BPMN 2.0 standard, data objects, data stores, data inputs, and data outputs.

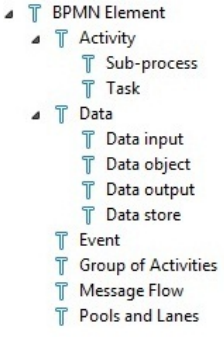


Fig. 5. Taxonomy of BPMN elements

The BPMN elements are annotated with security annotations, as illustrated in Fig. 6. Each security annotation applies to at least one BPMN element.

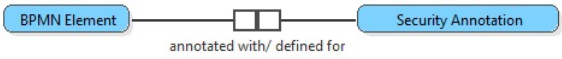


Fig. 6. BPMN element and security annotation concepts

The concept of security annotation is illustrated in Fig. 7.

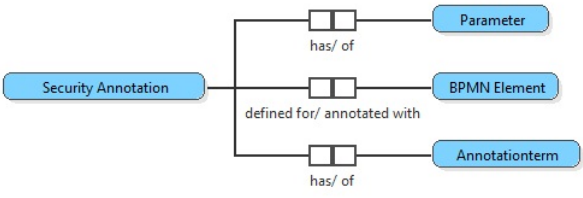


Fig. 7. Representation of the security annotation concept

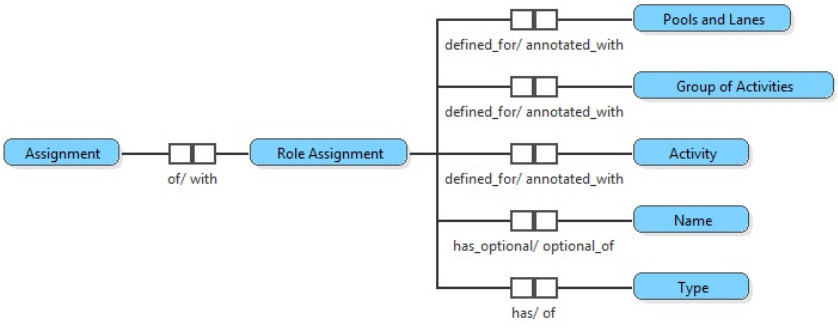


Fig. 8. Representation of the "role assignment" security annotation

Fig. 8 shows an example for the security annotation "role assignment" (annotation term, parameters, and its domain of BPMN elements are defined).In the authorization context, a role assignment specifies which role holders have to perform the annotated object (activities, group of activities, or all activities of annotated pools and lanes). The compulsory parameter of this type of annotation is "type"; "name" is an optional parameter. "Assignment" represents the annotation term of the "role assignment" security annotation.

The security constraints ontology is represented by the DOGMA ontology.

4.3 Ontology-Based Data Matching

Multiple ontologies of security concepts exist in TAS[3]. In this approach, we allow the process modeler to specify his/her queries by using generalized concepts from these

ontologies (seen as hierarchies, due to the domination relation specific to the security domain). This implies an ontology-based data matching between a concept from a general security ontology or from a user dictionary and the ontology of security constraints, used for specifying the security annotations.

Prior to the annotation, the system performs an ontology-based data matching step. This implies:1) mapping data into semantic networks (Tree, Directed Acyclic Graph/lattice or any directed graphs); 2) performing semantic computation, such as path recognition (shortest path, connectivity), path strength in scores (e.g., semantic vicinity), composite semantic similarity of semantic networks; 3) performing literal computation, such as fuzzy similarity of literals (strings); and 4) performing lexical computation, such as synonymous similarity (based on WordNet[6]) and similarity based on a user dictionary.

The searching task is performed via two modules: the interpreter and the comparator (as shown in Fig. 9).

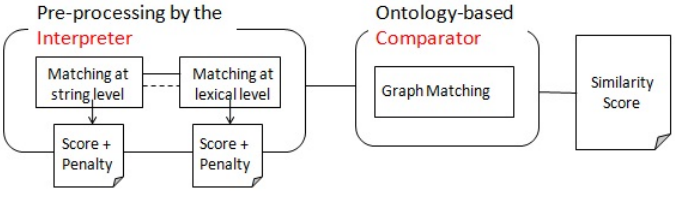


Fig. 9. Ontology-based data matching model

The interpreter makes use of a lexical dictionary and of the domain ontology to interpret the input term(s). Given a term that denotes a concept in the domain (security) ontology, the interpreter returns the correct concept defined in the ontology. The comparator then uses any combination of the different available graphs algorithms for the path recognition between two concepts originating from two different ontologies (or from a dataset and the ontology of security constraints). Currently the annotator searches for an exact match of a pattern, allowing similarity "1" (i.e. equality) only. Once the target concept is found, the annotation process is ready to start.

4.4 The Knowledge Base

The basic data element used by the knowledge annotator is represented by the Security Annotation Term, Element, Parameter, Value (STEPV) object. The STEPV object encapsulates the four entities needed to completely define a security annotation: the security constraint, the BPMN element being annotated, the parameters and their corresponding values.

For every security annotation that the process modeler intends to define, he must indicate as many fields (STEPV elements) as possible (according to his/her knowledge) corresponding to what he/she has in mind (see example from Fig.3). The process

[6] http://wordnet.princeton.edu/

modeler input is captured by annotator and analyzed in order to make recommendations to the process modeler. After performing the analysis, the system returns to the process modeler the STEPV elements that are related to his/her initial (usually incomplete) specification. The STEPV elements returned are considered valid annotations according to the constraints defined in the commitment layer (see Section 3).

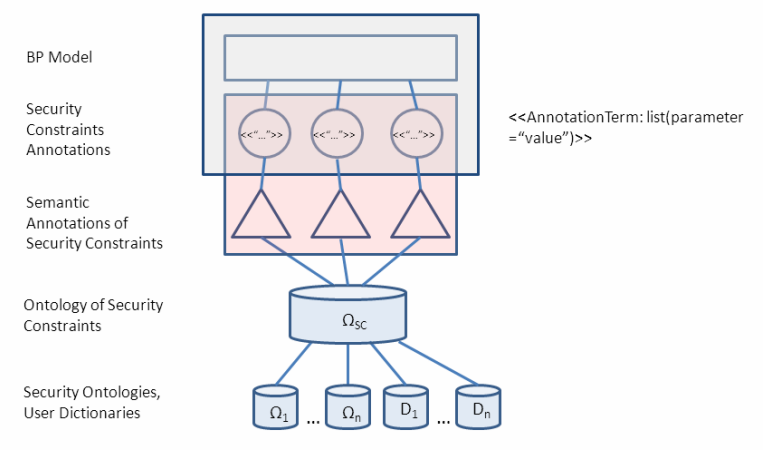


Fig. 10. Semantic annotation of security constraints

Once a STEPV object is complete and correct, our system stores it in the knowledge base. It will be used for later knowledge retrieval when the process modeler queries the system for recommendations. Note that the first three entities in every possible STEPV object (i.e., annotation term, BPMN element and parameters) are already stored in the ontology base, together with the relations they share with respect to one another (see Section 4.2).

When values are associated to parameters for a specific security annotation defined by the process modeler, we say that the security constraints ontology for that particular security annotation is instantiated. It is at this point that the knowledge base is capturing and storing the process modeler's knowledge (the way the process modeler chooses to instantiate the security annotation).

In order to ensure semantic interoperability between different organisations and actors regarding security concepts, we add an extra layer to the knowledge base, that is, the semantic annotations (as shown in Fig.10). Semantic annotations are added to the STEPV elements from different security-related ontologies or user dictionaries. The approach is explained in Section 4.3.

4.5 Knowledge Annotator Architecture

The knowledge annotator system was designed to assist the process modeler in designing security-annotated business processes with a user-friendly interface. Several functions are encapsulated in a web service, supported by six architectural components: A) capturer; B) annotator; C) indexer; D) retriever; E) comparator; and F)

presenter. The architecture is illustrated in Fig. 11. The annotator is implemented as a multi-tiered application, using J2EE[7] and Jena2[8] for reasoning.

The input to the web service call is collected by the UI. It is a query combining different terms expressing the process modeler's modeling intentions (such as "Request to user" AND "Activity" in Fig. 3). The result of a query sent to the knowledge annotator will be a set of correct security annotations proposed to the user via the UI. The added value of our approach is that the system allows query terms originating from multiple sources (e.g., user dictionary, security ontologies) which are mapped to concepts of the ontology of security constraints. This is realized prior to the actual annotation process, via an ontology-based data matching operation explained in Section 4.3.

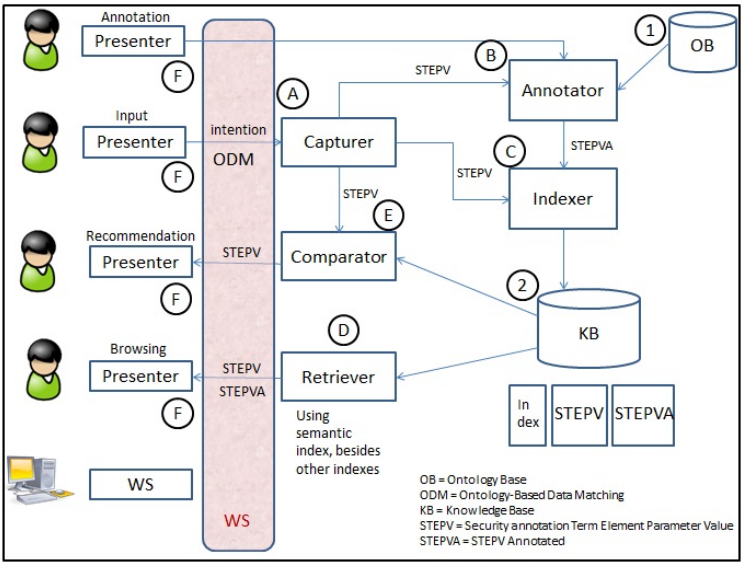


Fig. 11. Knowledge annotator architecture

The *Capturer* component captures the process modeler's intent and transforms it into an STEPV object. The STEPV objects are passed from one component to the other along the process (e.g. from the capturer to the indexer, from the indexer to the knowledge base, from the knowledge base to the retriever and from the retriever to the presenter). The process modeler specifies its input via a user-friendly interface. The UI presents a template with several fields for the process modeler to textually specify his security constraints. At this stage, the process modeler can be assisted by an ontology browser.

The STEPV object is annotated in a semi-automatic manner with concepts from the ontology base (OB), using the *Annotator* component.

[7] http://java.sun.com/j2ee/overview.html
[8] http://jena.sourceforge.net/inference/

The *Indexer* is used for the indexing of the STEPV elements stored in the knowledge base. This will facilitate the retrieval operation.

The *Retriever* component retrieves similar fragments (STEPV objects) from the knowledge base (e.g., all existing security annotations which share at least one common STEPV element with the input STEPV object). The similarity measure can be defined according to the user needs. For example, the user could only be interested in STEPV objects with a particular value for the "name" parameter. The knowledge base contains semantic annotation instances (STEPVA objects) of the security constraints.

The *Comparator* component performs a matching in order to compare the process modeler's demand (STEPV input object) with the resulted STEPV elements retrieved from the knowledge base in the past step.

Finally, the *Presenter* displays the user recommendations based on the design fragments (STEPV objects) retrieved from the KB. It is also the place where the user defines his/her queries. This component interacts with the process modeler via the UI. The WSs interaction is intended to provide interoperability in case of collaborative annotations done by members from different organizations. It implies human-system interactions at each organization end.

The following section shows how the knowledge annotator was applied in order to model secure business processes in an employability use case scenario.

5 Use Case Scenario

Within the TAS[3] project, we have developed an employability demonstrator focusing on the management of internships and work placements for university students [1]. Timely and accurate presentation and secure exchange of verified skills data and personal data is key factor to the success of this scenario. For example, recruiters and prospective employers want to access verified data in a standardized format (e.g., HR-XML[9]) to facilitate matching of students with job profiles. Similarly, candidates want to retain control over how their personal data is accessed, processed and stored by third parties.

Fig. 12 illustrates one of the employability scenarios in TAS[3]. In this scenario, Betty is a student at a UK university and seeks a work placement. Betty contacts a placement service approved by the university to discuss the details of her application. Her placement advisor, called Paul, informs her that he first needs to verify that she is a registered student at the university. Once Paul has received the confirmation, he contacts Betty to get permission to access relevant personal data to match her to available placements. Betty agrees to share her data provided that the data is not being shared to third parties without her approval. Based on this information, Paul identifies a number of placement providers that he believes to have suitable placements for students like Betty. Betty wishes to be put forward for two placements and agrees that the placement advisor can act on her behalf and she consents to have relevant personal data to be disclosed to them. Paul forwards Betty's personal data to the placement providers for consideration.

[9] http://ns.hr-xml.org/schemas/org_hr-xml/3_0/

There are several security constraints embedded in this process. Our goal is to technically realize this scenario as a business process and to use system support particularly for enforcing such security constraints. E.g, to check Betty's admission to the placement application, an identity provider component has to identify and authenticate Betty at the beginning of the process. In the same way, authorized access to Betty's personal data requires to specify access rules and to enforce these constraints by security components. To this end we model the placement process and descriptively annotate security constraints. A secure business process management system will enforce these constraints during process execution, i.e., in our example scenario, when Betty performs her placement application.

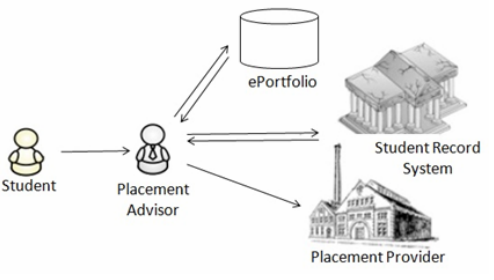


Fig. 12. The employability scenario

We now show exemplarily how security constraints of the employability scenario can be annotated efficiently using our tool. Fig. 13contains a business process activity "call matching service" of this scenario. It represents the matching of the personal data with placements available. The activity is annotated with three security annotations. In our example we use trust policies for the selection of adequate web services (i.e. placement providers) to be called. To this end, a modeler may start with an annotation "RequestToUser: SelectTrustPolicies". The annotator will find an annotation in the knowledge base, in particular a "RequestToUser: ServiceSelection" annotation which denotes that the student should be allowed to interactively select a web service. This is a typical involvement of users in an environment where trust policies for web services determine their use. The execution of a service discovery results in a list of web services adequate not only with respect to the required functionality but also to the trust level demanded by the caller. Additionally, the annotation "RequestToUser: SetDataPolicy" introduces a user interaction to set the data access policy for the user's personal data. The secure BPMS employs this policy when calling the matching service.

Analyzing this scenario, we identified the following use cases:

Use case 1. The first situation represents the basic use case for the knowledge annotator, when the process modeler is interrogating the ontology base (components A, B, D, F and 1 in Fig. 11) to retrieve and browse the correct concepts he/she needs in order to specify security constraints. The process modeler can ask the system to make faceted search on the BPMN elements (e.g., which annotations are defined for activities?), on the particular security annotation types, and on the parameters. If the

process modeler decides to annotate pools and lanes with the "Roleassignment" security annotation to specify that both, lane "Student" and lane "Placement Advisor" represent roles (see Fig. 4), the result looks like:

Table 1. Results returned for the faceted search: Roleassignment

SecurityAnnotation	BPMN Element	Parameters
Roleassignment	Pools&Lanes, Activities, Group of Activities	Type Name

Following these options, the process modeler either decides to make a choice or launches new queries in case the results do not correspond to his modeling intentions.

Use case 2. The second case represents the situation when the process modeler needs to instantiate the security annotation, i.e. to give values to the parameters of the annotation. In this situation, the system is performing matching operations between the process modeler input and the knowledge base content in order to retrieve instantiated security annotations of the same type. All components in Fig. 11 are involved.

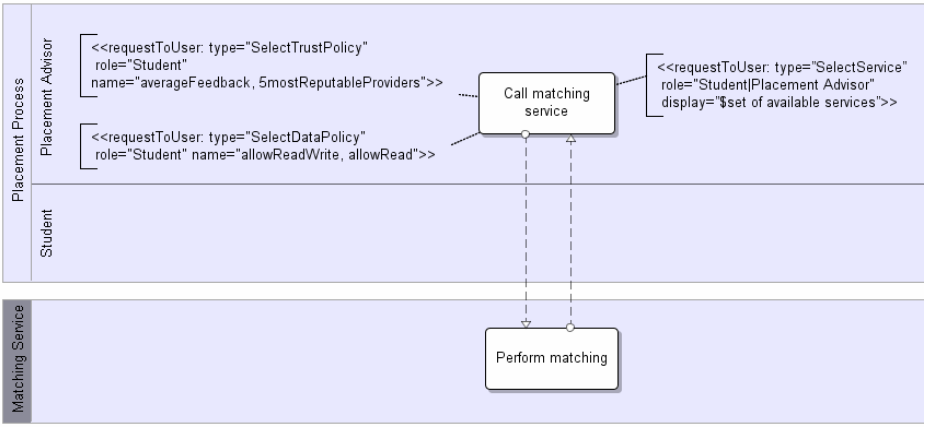


Fig. 13. Example of an annotated BPMN activity

Use case 3. The third use case is represented by the situation when the process modeler has created the security annotation by memory and needs to check for its syntactical correctness. In this case, the system performs similarity measures between the input and the ontology base and presents to the process modeler recommendations. All components in Fig. 11 are involved, except component 8 (i.e. the knowledge base).

Use case 4. The fourth situation is when the process modeler asks the system to check an instantiated security annotation for correctness. In this case the matching is done at the knowledge base level (matching performed not only at the ontology (type) level, but also at value level). The difference between use case 3 and use case 4 is that in the first case the correctness is validated only with the ontology base (syntax only), while

for the second case it is validated against the knowledge base. All components in Fig. 11 are involved.

Use case 5. The fifth situation represents the case when the actors are two web services which are interoperating via an interface. This situation represents the case when members of multiple organizations make collaborative annotations. The process modeler is involved in the annotation process at the organization end. All components in Fig. 11 are involved.

Currently, use cases 1 and 3 are completely supported. The implementation of other use cases is work in progress. We are actually working on enriching the knowledge base by domain experts via an online form and on the integration of a knowledge annotator user interface embedded in a BPMN editor.

6 Conclusion and Future Work

For specifying security constraints of business processes we have developed a security language for annotating process models. This paper presents a knowledge annotator which assists the process modeler during the specification of these constraints by providing semantic interoperability.

The knowledge annotator is designed to be (1) intuitive, acting as an intelligent system which is able to capture the process modeler modeling intentions and to provide him/her with design recommendations; (2) based on an ontology of security constraints grounded in natural language; (3) interoperable, being designed as a web service which operates in an open, distributed and dynamic environment; and (4) secure, enabling query-only requests via SSL/TLS links.

An emerging work is the consolidation of the knowledge base with security annotations designed for the two TAS^3 pilots (employability and e-Health).

Future work will involve linking the ontology with a more general ontology of security concepts (upper common ontology), which already exists in the TAS^3 project. The purpose is to assist the process modeler with more abstract security concepts when performing the search, providing him/her with a hierarchy of concepts for exploration.

Another future work is to integrate the annotator with an ontology-based interoperation service in order to be able to accept organization-specific vocabularies which map to the security constraints ontology.

Future directions also include user studies, aiming to analyze the user context and behavior in order to provide him/her with improved design suggestions.

Acknowledgments. This paper is supported by the EC FP7 TAS^3 (Trusted Architecture for Securely Shared Services) project. The authors would like to thank all TAS^3 project partners for their contribution to the research.

References

[1] Claerhout, B., Carlton, D., Kunst, C., Polman, L., Pruis, D., Schilders, L., Winfield, S.: Pilots Specifications and Use Case Scenarios, TAS3, Deliverable D9.1, Trusted Architecture for Securely Shared Services (2009), `http://tas3.eu/`

[2] Jenz, D.E.: Ontology-Based Business Process Management: The Vision Statement. Jens & Partner GmbH, 1st edn. (2003), http://www.bptrends.com/publicationfiles/12-03%20WP%20BP%20Ontology%20Vision%20-%20Jenz.pdf
[3] Hepp, M., Leymann, F., Domingue, J., Wahler, A., Fensel, D.: Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. In: Proceedings of the IEEE IECBE 2005, Beijing, China, pp. 535–540 (2005)
[4] Wetzstein, B., Ma, Z., Filipowska, A.: Semantic Business Process Management: A Lifecycle Based Requirements Analysis. In: Hepp, M., Hinkelmann, K., Karagiannis, D., Klein, R., Stojanovic, N. (eds.) SBPM 2007, Innsbruck, Austria (2007)
[5] Dimitrov, M., Simov, A., Stein, S., Konstantinov, M.: A BPMO Based Semantic Business Process Modelling Environment. In: Hepp, M., Hinkelmann, K., Karagiannis, D., Klein, R., Stojanovic, N. (eds.) SBPM 2007, Innsbruck, Austria (2007)
[6] Lin, Y.: Semantic Annotation for Process Models; Facilitating Process Knowledge Management via Semantic Interoperability. PhD Thesis, Norwegian University of Technology (2008) ISBN 978-82-471-5160-0 (printed version)
[7] Betz, S., Klink, S., Koschmider, A., Oberweis, A.: Automatic User Support for Business Process Modeling. In: Proceedings of the Workshop on Semantics for Business Process Management at the 3rd European Semantic Web Conference 2006, Budya, Montenegro, pp. 1–12 (2006)
[8] Born, M., Dorr, F., Weber, I.: User-friendly Semantic Annotation in Business Process Modeling. In: Proceedings of the Workshop on Human-friendly Service Description, Discovery and Matchmaking (2007)
[9] Di Francescomarino, C., Tonella, P.: Supporting ontology-based semantic annotation of business processes with automated suggestions. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) Enterprise, Business-Process and Information Systems Modeling. Lecture Notes in Business Information Processing, vol. 29, pp. 211–223. Springer, Heidelberg (2009)
[10] De Baer, P., Tang, Y., Meersman, R.: An Ontology-Based Data Matching Framework: Use Case Competency-Based HRM. In: Proceedings of the 4th Int. OntoContent 2009 Workshop, On the Move to Meaningful Internet Systems. LNCS, pp. 514–523. Springer, Heidelberg (2009)
[11] Spyns, P., Tang, Y., Meersman, R.: An Ontology Engineering Methodology for DOGMA. J. of App. Ontology 3(1-2), 13–39 (2008)
[12] Halpin, T.: Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design. Morgan Kaufmann, San Francisco (2001)
[13] Spyns, P., Meersman, R., Jarrar, M.: Data Modeling Versus Ontology Engineering. SIGMOD Record: Special Issue on Semantic Web and Data Management 31(4) (2002)
[14] Tang, Y., Meersman, R.: SDRule Markup Language: Towards Modeling and Interchanging Ontological Commitments for Semantic Decision Making. In: Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches. IGI Publishing, USA (2009) ISBN: 1-60566-402-2
[15] Mülle, J., von Stackelberg, S., Böhm, K.: A Security Language for BPMN Process Models. Technical Report, Karlsruhe Institute of Technology (KIT), no. 2011-09, Karlsruhe (2011)
[16] Mülle, J., Müller, J., Haberecht, T., von Stackelberg, S., Ciuciu, I., Reul, Q., Hoppenbrowers, J., Blandin, A., Boisvert, A.: Design of a Semantically underpinned, Secure & Adaptable Process-Management Platform, TAS3, Deliverable D3.1 (3rd iteration), Trusted Architecture for Securely Shared Services (2010), http://tas3.eu/

Workflow Support for Mobile Data Collection

Peter Wakholi, Weiqin Chen, and Jørn Klungsøyr

University of Bergen, POB 7802, N-5020, Bergen, Norway
peterokhisa@gmail.com
http://www.uib.no/infomedia/en/

Abstract. Mobile devices are increasingly being used for electronic data collection in low resource setting, where Internet-based solutions are infeasible. The data collection effort often requires the underlying processes, represented as data flows and workflows to be adhered. Workflow Management Systems (WFMS) could enable Mobile Data Collection (MDC) with workflow support as it is in Process Aware Information Systems. However, the use of WFMS for MDC designed for low resource settings needs to address challenges of mobile computing such as disconnections, slow connection links, limited computing power, etc. We present a framework that has been developed to integrate generic Data Collection tools with Workflow Management Systems (WFMS) to enable MDC in such resource-constrained environments. Furthermore we implement a tool based on this framework and provide an example of a vaccination registry project that uses mobile phones to record and track child immunisations.

Keywords: mobile data collection, process aware information systems, workflow management systems, mobile health.

1 Introduction

Mobile data collection has gained increasing prominence in low resource setting, because it enables instant digitization of data, hence saving time as compared to paper-based routines. The process of data collection can be considered to be a workflow. According to [13] data can be categorised in three types: non-time dependent, time dependent and cumulative data. Non-time dependent data is the data collected at a snapshot in time. Time dependent data is data collected repeatedly over time through multiple visits. Cumulative data is data collected over time but not linked to a specific visit. Time dependent and cumulative data have process-related activities and can be regarded as workflows.

MDC in mobile/wireless computing environment posts new challenges such as disconnections, slow connection links, and limited computing power which must be addressed when designing a mobile application. A workflow solution for MDC should take into consideration these characteristics and limitations by employing new computing paradigms. Several models have been proposed by [9] to support workflow in mobile environments. These include the client/agent/server, the client/intercept, the peer to peer, and the mobile agent. The reference model

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 299–313, 2011.

by the Workflow Management Coalition [8] specifies a framework for WFMS, identifying their characteristics, functions and interfaces and therefore provides a standard for implementation of generic workflows in MDC applications.

MDC applications like Cell Life, Dimagi, EpiSurveyor, GATHER, Open Data Kit, and openXdata are based on agreed open standards under Openrosa Consortium [12]. They provide generic, modular and extensible platforms for data collection and use common standards and tools for design of forms to collect data, rendering that form on a mobile device and storing collected data on a server to analyze. These standards are widely recognised in the MDC community [2]. In order to enable MDC in disconnected environments, questionnaire forms are downloaded to the mobile device. When forms have been appropriately filled, they are uploaded to a central server when connection is established.

In this paper, we propose a framework for the integration of a WFMS based on YAWL [24] with a generic MDC tool called openXdata [10]. We describe the design and implementation of a workflow adapter that acts as a bridge between the mobile device, data processing applications and workflow engine. Through this implementation, we have been able to provide a distributed architecture that enables the ordering of tasks linked to mobile devices and web-based applications. In addition, we provide an example where this framework has been used in a vaccination registry that uses mobile devices to collect data on child immunisations.

This paper proceeds as follows. Section 2 describes an example scenario for mobile data collection in child vaccination process and discusses the challenges and requirements for designing workflow support for MDC. Section 3 Provides a conceptual framework that describes how workflows can be implemented in a mobile environment. Section 4 describes the design and implementation of the system for the example scenario. Section 5 provides a theoretical and comparative analysis of the implemented solution relative to workflow patterns implemented in YAWL. Related work and the contribution of the paper are discussed in section 6. Finally, section 7 provides future work and further discussions.

2 Example Scenario

In this section we present a working example of mobile data collection in an m-health project implementing an immunisation registry (MVAC) to illustrate the workflow and dataflow in the MDC process. The MVAC project [11] aims to improve immunization programs through the use of mobile phone based recording of child vaccination and health data in immunization registers. The work described in this paper part of the design and implemention of the system.

The current immunisation process requires children and their caretakers to visit a health center at least 4 or 5 times in the first year of life. The typical activities around immunization of one child in a paper-based registration system can be represented in a simplified workflow diagram shown in figure 1. A new child enters the public health system. This can be a newborn or a child migrating from another area, for which no records exist locally. Personal information for

the child and its caretaker is registered. If this child already has an immunization history (as per the mother's recollection or child immunization card), this is also registered in the local registries. The child is immunized if the nurse established it is due for immunization. The immunization is registered: usually either by a tick in the registry book, or indicating the date of the vaccination. An appointment is made for the next visit, mostly informally. The nurse prepares for immunization, mostly on a monthly basis. That involves always ordering vaccines, scheduling sessions and sometimes reminding the caretakers of the appointments.

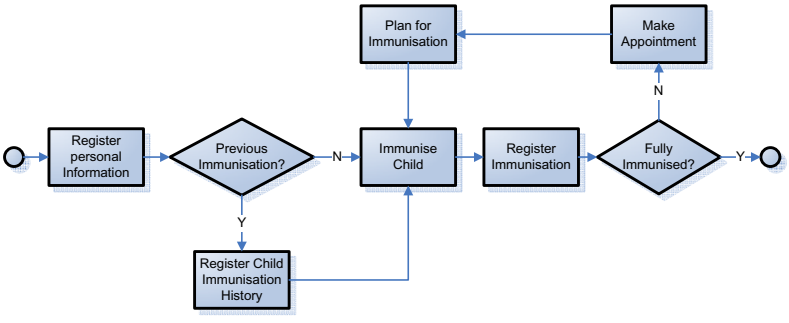


Fig. 1. Activities in paper-based immunization process

The vaccination process described above is implemented in Figure 2 using the YAWL language and Workflow Management System [21]. In this implementation, when a caretaker and child visit a health facility, the health worker uses a mobile phone to check if they are already registered in the vaccination registry. If the caretaker or child does not exist, they should be registered. If the child exists, the schedule is checked to ascertain if he/she is due for immunisation. If immunisation is not due, a Short Message Service (SMS) message for the next appointment is sent to the mother and the process ends. After registration or confirmation of scheduled visit, the child is immunised and an immunisation form completed on the mobile.

The process described above represents typical workflow and dataflow scenarios in MDC. In order to provide a definition of a workflow, the following need to be specified: control flow definition, data definitions, resource classification and resource management rules [22]. In the control flow definition, a description of the process itself is defined by routing of work as tasks are undertaken to meet a business process. Figure 2 shows, the control flow definition by defining the ordering of tasks. The resource classification and resource management rules provide the classification of the resources to be used and how to map work to resources. In the perspective of the system, the mobile device represents the resource that will execute a task. The data definition provides for the routing information for each case and is specified at design time. In MDC, data is collected based on a study and thus provides the routing data. A study is composed of a set of forms, each form having one or more questions.

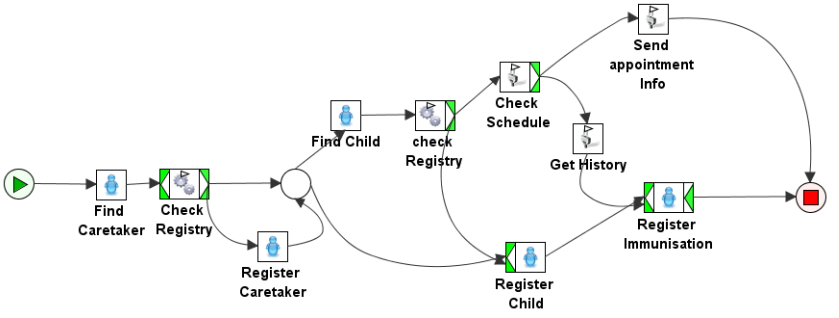


Fig. 2. implementation using YAWL and mobile forms service

[5] provide a widely acclaimed framework to understand the context and identify potential usability issues for a variety of software development projects. This framework is also a useful tool for articulating the issues relevant to MDC implementing workflows. The framework includes six profiles that are used to understand the context: environment, device constraints, information, interaction, incumbent, and operational risk profile. Environment deals with the factors surrounding the context of use. Device constraints are the hardware and software technology decisions and affect the design of the system. Information profile considers the volume, complexity, source and flow of the data being collected.The information profile therefore relates to the data that is sent and the information received by the user. In a business process, it is provided by data definition. The incumbent profile provides for users, their work and the context in which they work. The resource classification and management rules in business process provides for this. Finally interaction profile presents the manner in which a user interacts with the system which is provided by the control flow definition.

The environment and device constraint profiles relate to the mobile networks and devices used. These have been extensively discussed in section 1 as forming the basis for this research. Based on these usability profiles, the following usability issues were identified for MDC based on the MVAC project:

1 *Data definition/information profile:* Forms used in MDC need to be mapped to tasks in a workflow specification. The elements of a form are questions and these need to be mapped to task parameters. The visibility and scope of task parameters should be made available in the question. In a study, data that needs to be viewed when performing a task (e.g. vaccination history) is pre-filled in the corresponding questions.
2 *Resource definitions/incumbent profile:* WFMS often implement client-server architecture in a connected and reliable environment where work lists are sent to the resource (client). The traditional client-server model used by many WFMS needs to be replaced by more appropriate models as argued by [1] [9].
3 *Control Flow definition/interaction profile:* A mobile and disconnected environment would pose challenges in enabling the implementation of some

control flow constructs. The challenge is to provide a solution that allows for a variety of these constructs to be executed by a resource using a mobile client.

4 *Workflow Life cycle:* The workflow life cycle is concerned with the states that a work item goes through when it is created. These states need to be supported by the implemented solution. In addition, enabling the launch of a new case from a mobile device thus creating the first work item in a process is required.

In order to address these issues, a conceptual framework that allows for Integration WFMS into MDC was developed.

3 The Conceptual Framework

Conceptual modeling is a key stage is workflow system development. This approach is enhanced by the ability to carry out a simulation of the model prior to development. Coloured Petri nets (CPN) tools provide a good basis for developing Conceptual frameworks for workflows [19] and were used to modeling the life cycle of a work item in a mobile environment. The model enabled multiple simulations to be done so as to refine the ideas proposed. The goal of this model is to provide the formal specification for deploying WFMS in a MDC environment.

The conceptual framework illustrates the management of work items and resources in executing activities in a mobile environment. Figure 3 shows the conceptual framework. A work item handled by the form service is checked out and mapped to the corresponding data collection form. It is then assigned to appropriate resource represented by the mobile agent on the server. Once it is downloaded to the mobile device, it is in the 'executing' stage and cannot be assigned to another resource. Upon completion of the activity, the relevant case data is extracted by the form service and the work item is checked in and the cycle ends. The model further allows for mobile users to initiate new cases. The process for initiating a new case starts with the mobile device component. The following components are required; *workflow engine, launch case, forms service, mobile agent* and *mobile device* (client application). These are described in details below:

A *Workflow engine*
The WFMC reference model [8] provide workflow enactment service as *"software that interprets the process description and controls the instantiation of processes and sequencing of activities, adding work items to the user work lists and invoking application tools as necessary"*. This is done through one or more co-operating workflow management engines. In the Conceptual framework, the workflow engine keeps track of the current status of the work items and generates new work items according to a predefined process model.

B *Launch Case*
The ability to start a new instance of a process is a key requirement for a mobile user. The WFMC reference model provides for workflow client

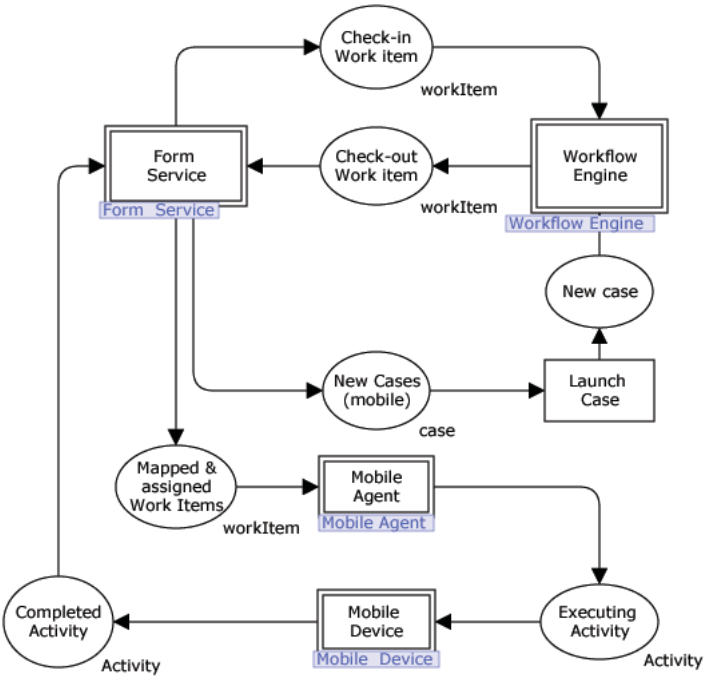


Fig. 3. Conceptual framework showing interaction of major component in executing a work item

application interface with the ability to create or start and termination of an individual process instance. In the framework, this is achieved by allowing the form attached to the first task in a process can be used to launch a new case. When a user fills this form, the case data provided is used as input parameters for the process model and the first task mapped to the form.

C *Form Service*

The WFMC reference model provides for Service-oriented Architecture where workflow enactments services are called up by a task to provide functionality that implements the service [8]. The implementation of a form service further provides a framework for mapping workflow tasks and generic forms at the enactment stage. The form service also manages the resource allocation, where work lists are allocated to the designated resource.

Figure 4 shows an entity relationship diagram that illustrates the mappings. There exists a hierarchical relationship between the entities of a study and workflow. In the mapping, each study is matched to one workflow. A task may be matched with one form whereas a form can be matched with many tasks. This is to enable the same form to be used at different stages of the process. The input and output variables of a task can be matched to questions in a form. Each variable can only be matched to one question and vice versa.

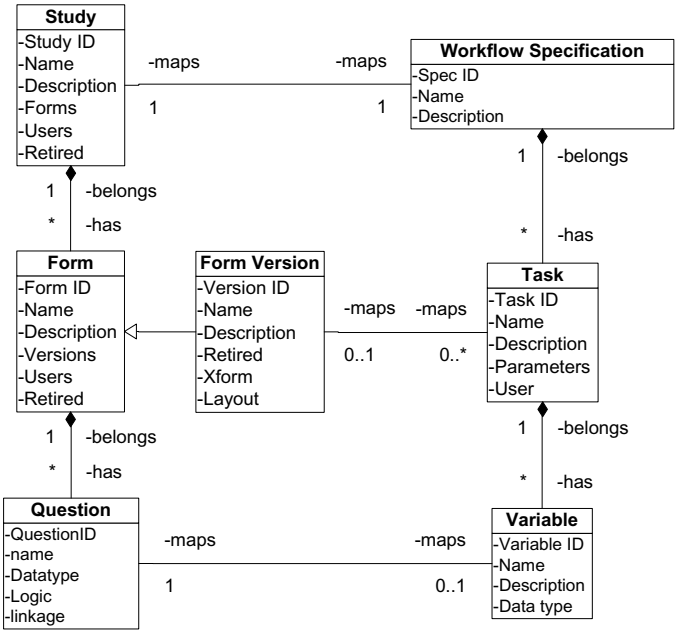


Fig. 4. ERD showing the mapping of studies to workflows

As illustrated in Figure 3, the form service receives work items from the engine, maps them and passes to the mobile agent. The form service also receives data from the mobile device and checks in the work item once execution is complete. To allow for launching of processes from the mobile device, work items for the first task are automatically created once linked to the form. If this form is completed, the form service launches the process and then the *check-out* and *check-in* steps are completed with the filled data.

D *Mobile Agent and Client*

As previously discussed, the traditional client/server model used by workflow systems does not provide a viable model for implementation in a mobile environment. The mobile client needs to fulfill its responsibilities of executing work items in a disconnected environment as if it were always available. In the Conceptual framework, we propose to achieve this through a mobile agent, by implementing a client/agent/server model as proposed by [3]. It uses messaging and queuing infrastructure for communications from the mobile client to the agent and from the agent to the server. In our implementation, the agent receives all work items on behalf of the mobile client. When connection is established, these items are transferred to the mobile client.

4 System Implementation

In this section we discuss the implementation of the system based on the Conceptual framework and illustrate its application in the immunisation registry. We begin by discussing the mapping between workflows and studies, the architecture of the implemented system (workflow adapter) and then illustrating the MVAC project prototype.

4.1 Mapping Workflows to Studies

In order to attain the mapping described in the previous section, a database that keeps all mapped studies and workflows was created. The details of the mappings between forms and tasks and questions and variables were stored as XML files as illustrated in Figure 5. XML was used because it provides a flexible way of storing data not limited to predefined attributes as it is in relational databases. These mapping are based on the YAWL WFMS and openXdata tool for MDC.

The schema shows the top level mapping where a workflow specification (*vaccination.yawl0.43*) is mapped to a study (*vaccination*). The next level in the hierarchy is mapping tasks (e.g. *register_ Child_ 7*) to forms. For each task, a form is mapped to it. A form may have many versions, so every task should be mapped to a form version as well (e.g. *register_ Child*) is mapped form version of ID 16). The last level in the hierarchy, maps questions in a study to input and output parameters of the workflow. In the example, we have seven input and output parameters of the task *register_ Child*, mapped to seven questions in the openXdata form. The mappings take into consideration the data types to ensure that the right type of data is exchanged between the WFMS and the MDC tool.

```
<Maps>
    <SpecStudyMap Id="15" SpecId="vaccination.yawl0.43" StudyId="Vaccination">
        <TaskFormMaps>
            <TaskFormDef TaskId="Register_Child_7" FormId="11">
                <TaskFormVersionMaps>
                    <TaskFormVersion TaskId="Register_Child_7" FormVersionId="16">
                        <ParamQuestionMaps><ParamQuestion Question="caretaker" Parameter="caretaker"/>
                            <ParamQuestion Question="birth_place" Parameter="BirthPlace"/>
                            <ParamQuestion Question="mother" Parameter="mother"/>
                            <ParamQuestion Question="national_id" Parameter="nationalID"/>
                            <ParamQuestion Question="gender" Parameter="Gender"/>
                            <ParamQuestion Question="last_name" Parameter="lName"/>
                            <ParamQuestion Question="notes" Parameter="notes"/>
                            <ParamQuestion Question="appointment_id" Parameter="appoinmentID"/>
                            <ParamQuestion Question="first_name" Parameter="fName"/>
                            <ParamQuestion Question="father" Parameter="father"/>
                        </ParamQuestionMaps>
                    </TaskFormVersion>
                </TaskFormVersionMaps>
            </TaskFormDef>
        </TaskFormMaps>
    </SpecStudyMap>
</Maps>
```

Fig. 5. Sample XML mapping of workflow specifications and studies

4.2 Architecture of Workflow Adapter

The architecture of the system is described using components as illustrated in Figure 6. The Workflow Reference Model [8] consists of five interfaces; Process Definition Tools, Workflow Client Applications, Invoked Applications, Workflow Interoperability and Administration and Monitoring. The execution of these services is invoked through the Workflow Client Applications interface (Interface B in YAWL). Using this interface, the form service was implemented to manage the work items by invoking the services of the components for *resource management, specification management* and *work item management.*

Specification management ensures that tasks and variables are matched to the corresponding form and question mappings. The mappings are stored in the database as XML files as illustrated in Figure 5. *Resource management* ensures that tasks are assigned to the right resources for execution. When a work item is generated, *Resource Management* and *Specification Management* components are invoked to provide the necessary resource and questionnaire mappings before execution. This is done through the component *Work item management*, which then passes the request to the *mobile agent.* The *mobile agent* is charged with holding the work list for a defined mobile device until connection is established. Once the work list has been downloaded, it is appended to the appropriate questionnaire form for data collection. The *Stream Handler* ensures that the mobile device has the right forms for data collection and extracts the relevant information required for input to the next stage of the process.

4.3 Immunisation Project Prototype

The immunisation registry prototype implemented uses the workflow adapter for data collection. The process model shown in Figure 2 was used. The tasks *Find Caretaker*, *Register caretaker*, *Find child*, *Register Child*, and *Register Immunisation* are executed using mobile forms. Figure 7 provides a screen shot of the *Register child* task whose mapping to MDC forms was shown in Figure 5. The first screen shot shows a work item *register child* enabled, while the next screen shows the corresponding form for data collection that was opened with pre-filled data about the child to register.

Health workers are provided with mobile devices that have the openXdata application installed. When a child comes for immunisation, the health worker fills in the *Find Caretaker* form. When this form is received by the workflow adapter, it launches a new process and provides the input parameters. These parameters are passed by the YAWL engine to the next task which is an automated check of the registry for any matching information. Information about the Health Worker who made the request is captured by the system and a response is sent to the mobile agent work list. This illustrates the relevance of the *retain familiar* resource pattern. When the Health Worker asks for the next task by downloading the enabled work items, either the caretaker is available in the registry or not. The subsequent work items are enabled as described in section 2.

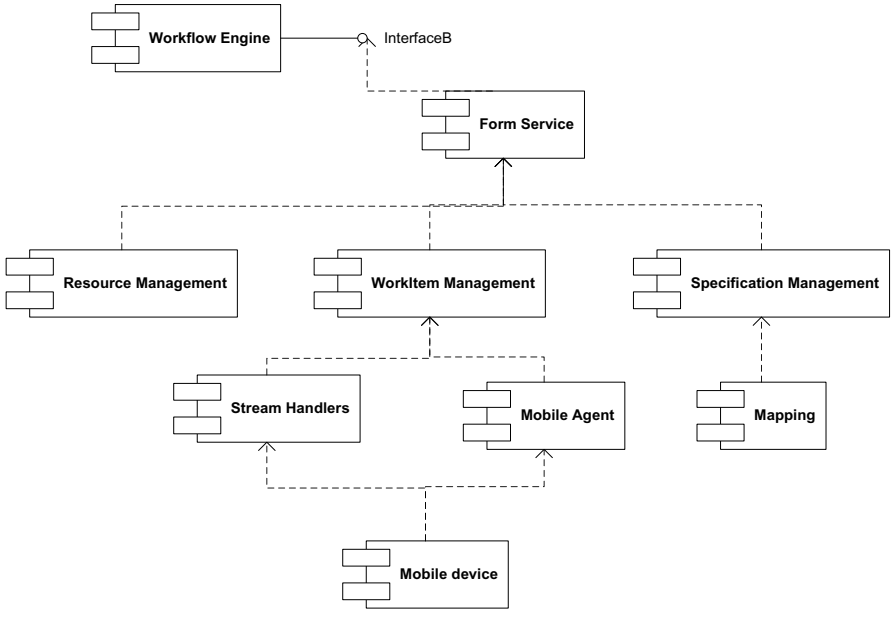


Fig. 6. overall architectural structure for the MDC workflow module showing top-level components

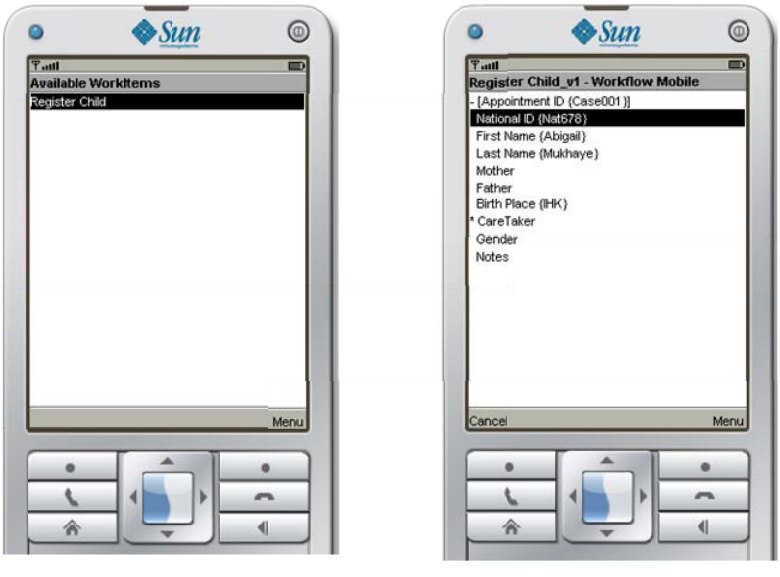


Fig. 7. implementation of the *Register Child* task on a mobile phone

5 Analysis of the Framework

The workflow adapter is a generic tool and therefore, we provide a theoretical analysis of the conceptual framework in order to determine its support for workflow patterns. Workflow patterns provide an ontological basis for understanding the expressiveness of the process modelling tools and languages [25]. In Table 1 we make comparisons of the implemented workflow adapter to existing YAWL web-based functionality. These patterns provide a control flow, resource management and data perspective which relate closely with the issues identified in

Table 1. Comparison of the workflow adapter with YAWL and workflow patterns

Workflow perspective	WFMS/YAWL Support	Workflow Adapter
Control flow	The YAWL system supports the implementation thirty onecontrol-flow patterns [21].	Cancellation when workitem is downloaded to mobile phone is not possible. Cancellations are done when connection is re-established.
Resource	[18] provides workflow resource patterns, 43 resource of which are supported by YAWL [20].	Supports only the *selection autonomy* and *retain familiar* patterns. Based on requirement between two tasks, that where possible they be executed by the same user and ability for a resource to choose which work item is undertaken next.
Data	Full support for the data perspective. Data elements are defined and used for conditional routing, for the creation of multiple instances, for exchanging information with the environment [23].	All the patterns related to data visibility, interaction and transfer and routing in YAWL are limited by the mapping of questions and parameters. This mapping requires that similar data types and structures of the data elements are matched. For this implementation, it is only possible to map *String, Integer, Decimal, Double, Date, Time, Datetime* and *Boolean* data types.
Work item lifecylce	[18] decribes the states of a work item as *offered, allocated, started, suspended, resumed, failed* and *completed*. These are fully supported by YAWL.	The failed state is not supported. All workitems allocated are expecetd to be completed as it is not possible to keep track of the state when the mobile device is disconnected.
Atomicity	A task which corresponds to a single unit of work is atomic. Other types include block, multi-instance and multiple-instance block [18]. These are supported by the YAWL system.	Only atomic tasks are supported. Mapping of such block, multi-instance and multiple-instance block to forms in a study causes complication as there is no equivalent representation of the underlying constructs.

section 2. Additionally, we make an assessment on the handling of work items from creation to completion and dealing with atomicity.

It can be observed that the workflow adapter has some limitations. However, these limitations are discounted by the fact that in MDC, complex constructs are seldom required. Typical contol flow requirements for MDC include; support of sub-forms (e.g. collect data on household and multiple household members); ability to conditionally link forms; ability to map data values between linked forms; which are provided by the current implementation. The resource constracts provided were limited to those necessary for the MVAC project - again only 2 of these constructs were neccessary. One key constraint is that mobile phones do not have uniques addresses making the *push* based patterns infeasible. The data constructs provided were sufficient to cover common uses. However, there is need to expand the possibilities of the various data types to capture more varieties used in MDC like voice, images, video, etc. To cater for the *failed* state, reassignment of incomplete tasks are done by the administrator. We also provided for flexibility to have a workitem to be selected by more than one mobile device to allow it be executed at the point of contact with a client (assuming earlier assignment to a different resource). Dealing with block and multiple instances should be explored in future work.

6 Related Work

This paper presents a framework for integration of WFMC into the mobile environment for data collection. This framework is validated through the design and implementation of the workflow adapter for MDC. In addition, we have presented a working example that gives a compelling case for workflow support for data collection. We draw knowledge from WFMS as based on the WFMC reference model [8], workflow patterns [25],[18] and standards for MDC provided by the openRosa Consortium [17],[12],[6].

Previous work on workflow support for mobile devices has concentrated on developing architecture for WFMS to enable execution of tasks on the mobile and developing a light-weight WFMS to work on mobile devices. [7] contend that to adapt mobile devices to support workflows, it requires a general extension of existing workflow systems by adding mobile device as one of the environments where activities can be executed. They propose a WFMS for mobile devices called SLIVER which is based on business process execution language (BPEL) [7]. In the implementation, a mobile phone acts a server for other mobile phones which are clients. Connection to the server assumes a connected environment and is achieved through Bluetooth and Wifi technologies. The approach used in creating a client for SLIVER and connection mechanisms were used in our implementation to develop an appropriate client with a small footprint for low-end mobile phones.

[14] provide a prototype based on Java Border Service Architecture (JBSA) in which mobile workflow users are connected to a central workflow engine via mobile devices. The prototype provides a possibility to use a Web browser, a

PDA, and even a WAP phone as the workflow client to receive and execute work items. This approach has been used to enable download and upload of workitems and while allowing the server deal with data processing and manipulation. Additionally, MDC tools based on Openrosa standards like Javarosa [12] do not explicitly implement workflows but are developing elements that are workflow related. This includes the ability to view pre-filled forms and link forms in a study. These have been used as a basis for requirements and motivation for workflow implementation for MDC.

Whereas the work done by [15],[14],[7] provide a means of executing work on a mobile phone, the work presented in this paper builds on that and proposes a framework for integration of such systems to meet the challenges of disconnected environments, low computing power and slow intermittent connections. We contend that the ideal solution for resource-contrained and unreliable mobile environments requires a more robust framework, hence the reason for the approach that this paper presents. So rather than presenting another architecture, we present a conceptual framework for providing an ideal solution. The ideas presented in this paper futher provide a generic approach to integrating MDC with WFMC through mapping studies and workflow specifications. It can be argued that the work presented in this paper provides a comprehensive, ontology and standards based approach to data collection processes which are time-dependent and cumulative.

7 Future Work and Discussion

A situation often arises when a mobile user is required to carry out a series of predefined tasks before a connection is established. The current system does not allow for this as a connection will always be required before the execution of the next task. It is therefore necessary that future work focuses on how workflow support can be achieved in such situations. The YAWL system provides a concept of worklets (group of related tasks) that could be exploited to assign more than one task to the mobile user.

There have been attempts to develop a framework that assigns multiple tasks to mobile devices in disconnected environments [4]. [16] propose a general approach to synchronise distributed workflows based on state and activity charts to give a status of a workflow process. Future work will involve developing these approaches further to support execution of work in mobile, disconnected and distributed environments.

The work presented in this paper sets a platform for further exploration of these issues in a fast developing and useful domain of MDC. Because MDC applications aim to collect data using low-end mobile phones in disconnected environments, the implementation of the system in the mHealth domain clearly illustrates the authenticity of the framework. The working example presented in this paper will be field-tested to determine its usability in an actual environment. Empirical evaluations of the MVAC prototype will be done to determine the usability of the tool. Lessons learned from these field tests will further enhance the research.

Acknowledgments

The authors would like to thank the openXdata team for the support in developing this work. Special thanks go to Ronald Kayondo for the work done in integrating YAWL to OpenXdata. We also thank the anonymous reviewers, whose comments helped to improve this paper.

References

1. Alonso, G., Gunthor, R., Kamath, M., Agrawal, D., Abbadi, A., Mohan, C.: Exotica/FMDC: a workflow management system for mobile and disconnected clients. In: Databases and Mobile Computing, pp. 27–45 (1996)
2. Anokwa, Y., Hartung, C., Lerer, A., DeRenzi, B., Borriello, G.: A new generation of open source data collection tools. In: 2009 International Conference on Information and Communication Technologies and Development (ICTD), p. 493. IEEE, Los Alamitos (2010)
3. Badrinath, B.R., Bakre, A., Imielinski, T., Marantz, R.: Handling mobile clients: A case for indirect interaction. In: Proceedings of Fourth Workshop on Workstation Operating Systems, pp. 91–97. IEEE, Los Alamitos (2002)
4. Bahrami, A., Wang, C., Yuan, J., Hunt, A.: The Workflow Based Architecture for Mobile Information Access in Occasionally Connected Computing. In: IEEE International Conference on Services Computing, SCC 2006, pp. 406–413. IEEE, Los Alamitos (2006)
5. Constantine, L.L., Lockwood, L.A.D.: Software for use: a practical guide to the models and methods of usage-centered design. ACM Press/Addison-Wesley Publishing Co., New York (1999)
6. Curioso, W.H., Mechael, P.N.: Enhancing'M-Health'With South-To-South Collaborations. Health Affairs 29(2), 264 (2010)
7. Hackmann, G., Haitjema, M., Gill, C., Roman, G.C.: Sliver: A bpel workflow process execution engine for mobile devices. In: Service-Oriented Computing–ICSOC 2006, pp. 503–508 (2006)
8. Hollingsworth, D., et al.: Workflow management coalition: The workflow reference model. Workflow Management Coalition (1993)
9. Istepanian, R.S.H., Pattichis, C.S.: M-health: Emerging mobile health systems. Springer-Verlag New York Inc., New York (2006)
10. Keenan, A.: Remote Mobile Data Collection. PhD thesis, University of the Western Cape (2010)
11. Klungsøyr, J., Grevendonk, J.: Using mobile phones to track immunizations zations (October 2010), http://www.uib.no/cih/en/nyheter/2010/10/using-mobile-phones-to-track-immuni
12. Klungsøyr, J., Wakholi, P., Macleod, B., Escudero-Pascual, A., Lesh, N.: OpenROSA, JavaROSA, GloballyMobile–Collaborations around Open Standards for Mobile Applications. In: M4D 2008, General Tracks (2008)
13. Moon, K.K.: Techniques for Designing Case Report Forms in Clinical Trials. ScianNews Volume, DOI (2006)
14. Muller-Wilken, S., Lamersdorf, W.: Jbsa: An infrastructure for seamless mobile systems integration. In: Trends in Distributed Systems: Towards a Universal Service Market, pp. 164–175 (2000)

15. Muller-Wilken, S., Wienberg, F., Lamersdorf, W.: On integrating mobile devices into a workflow management scenario. In: Proceedings of 11th International Workshop on Database and Expert Systems Applications, pp. 186–190. IEEE, Los Alamitos (2002)
16. Muth, P., Wodtke, D., Weissenfels, J., Dittrich, A.K., Weikum, G.: From centralized workflow specification to distributed workflow execution. Journal of Intelligent Information Systems 10(2), 159–184 (1998)
17. Openrosa. Openrosa consortium (February 2011), http://openrosa.org/
18. Russell, N., Ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow resource patterns. Citeseer (2005)
19. Salimifard, K., Wright, M.: Petri net-based modelling of workflow systems: An overview. European Journal of Operational Research 134(3), 664–676 (2001)
20. Tan, H., van der Aalst, W.: Implementation of a YAWL work-list handler based on the resource patterns. In: 10th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2006, pp. 1–6. IEEE, Los Alamitos (2006)
21. ter Hofstede, A.H.M., van der Aalst, W.M.P., Adams, M.: Modern Business Process Automation: YAWL and its support environment. Springer-Verlag New York Inc., New York (2009)
22. van der Aalst, W.M.P.: Business process management demystified: A tutorial on models, systems and standards for workflow management. Lectures on Concurrency and Petri Nets, pp. 21–58 (2004)
23. van der Aalst, W.M.P., Aldred, L., Dumas, M., ter Hofstede, A.H.M.: Design and implementation of the YAWL system. In: Advanced Information Systems Engineering, pp. 281–305. Springer, Heidelberg (2004)
24. Van Der Aalst, W.M.P., Ter Hofstede, A.H.M.: YAWL: yet another workflow language. Information Systems 30(4), 245–275 (2005)
25. van Der Aalst, W.M.P., Ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and parallel databases 14(1), 5–51 (2003)

Adapted UML Activity Diagrams for Mobile Work Processes: Experimental Comparison of Colour and Pattern Fills

Sundar Gopalakrishnan, John Krogstie, and Guttorm Sindre

Department of Computer and Information Science
Norwegian University of Science and Technology (NTNU)
Sem Sælands Vei 7-9, 7491 Trondheim, Norway
{sundar,krogstie,guttorm.sindre}@idi.ntnu.no

Abstract. For multi-channel information systems it is often relevant to model where something is supposed to take place, but business process modelling notations seldom capture geographical location. In previous papers, we suggested and compared alternatives for small modifications to UML Activity Diagrams to address this, and a controlled experiment indicated that an alternative using colour performed better than one using annotations. However, colour also has some challenges, especially concerning users with colour vision problems. Hence, this paper reports on a new experiment comparing colour with black/white pattern fills. The experiment investigated both the participants' opinions about the notations and their performance on some tasks. While opinion was significantly in favour of the colour notation, task performance was only slightly in favour of this notation, and not significantly so.

Keywords: Requirements specifications, mobile information systems, model-based development, UML activity diagram, process modeling.

1 Introduction

Whereas location is discussed a lot in e.g. CSCW [1] mainstream process notations used in IS modelling tend to ignore the location aspect. For instance, BPMN [2] and UML activity diagrams [3] capture what (objects), how (sequence and parallelism of activities and decisions), who (swimlanes), when (time triggers and time events), and to a very limited extent why (e.g., how a decomposed activity diagram satisfies a higher level activity) - for the latter some extensions with process goals have also been suggested [4], but they do not capture the location of the activities performed. As long as work is performed by people sitting in their offices using desktop computers, the neglect of physical location is understandable - it is much more important whether a task is performed by the salary or purchasing department than whether the worker is sitting in office K42 or B88. Hence it is understandable why swimlanes are used to denote organizational placement rather than geographical placement.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 314–331, 2011.

For mobile and multi-channel information systems, however, the location and context of activities is important [5]. Whether a certain task should be performed in the office before going on a site visit, in the car while driving, after arriving at the site - or possibly any of these places, according to the employee's choice - could have a large impact both on quality, efficiency, worker satisfaction and customer satisfaction, and would thus be an important process design decision. In turn, this would have bearings on what IT solutions to use to support the work process, and what requirements these solutions would have to satisfy. For instance, if the tasks were to be performed while driving a car, this would cause other demands on usability than working in the office.

Hence, we want to investigate adaptations of mainstream process notations such as UML activity diagrams to also be able to capture the location of activities. A minor adaptation of, e.g., UML may be better than inventing an entirely new notation because industry is more likely to take up a notation which appears familiar. Relating to UML, there is also a long tradition in providing modelling profiles, having small extensions to the core notation [6] presented a number of notation ideas for including location in UML diagrams and compared them analytically. Some of the notations turned out to be clearly inferior, either lacking expressive power or expressive economy or becoming messy due to a high number of crossing lines. Two of the proposed notations came out as more promising than the others, one adding location/context by means of annotations (i.e. UML notes) and the other using colour. Subsequently, a controlled experiment was conducted to compare these two, where the participants (students) were asked to perform some tasks with the models (answering true/false questions and finding model defects), as well as give their opinion about the notations using a TAM-inspired questionnaire. This experiment, reported in [7], showed a significant advantage in task performance for the colour notation, while there was no significant difference in the opinion about the two alternatives. Although giving the best results with our participants, the colour alternative could cause problems for users with impaired colour vision. Hence, if a notation using black/white colour fills gives equally good results, this might be a good alternative to the colour notation. So, we performed a new experiment similar to the one in [7], but now comparing colour vs. black/white patterns, which is reported in this paper. The rest of the paper is structured as follows: Section 2 presents the result from our previous work and some related work in this area, section 3 discusses the research method, and section 4 presents the experiment results. Section 5 provides analysis and discussion, and section 6 concludes the paper. The appendix contains some material about the details of the experiment, for the especially interested reader.

2 Previous and Related Work

As already stated in the introduction, there was an initial analytical investigation [6] of some alternative notations, where two alternatives came out as more promising than the other, namely one notation using annotation (i.e., UML notes) to indicate where an activity took place, and the other used colour to indicate this. The analytical evaluation used frameworks as indicated in table 1 and 2. In addition to the notations shown in the table, some additional ones were also considered but evaluated as being poorer.

Table 1. Evaluation of proposed notations with simple and large models

Notation	Min. deviation from standard		Expressiveness		Intuitive / Easy to read		ModelComplexity	
	Simple	Large	Simple	Large	Simple	Large	Simple	Large
Trad. UML	+ +	+ +	- -	- -	+ +	+ +	+ +	+ +
Annotated	+	+	+	+	-	- -	+	- -
Colours	-	-	+ +	+ +	+	+	+ +	+ +
Patterns	-	-	++	++	+	+	++	++

(++) more positive, (+) positive, (-) negative, (--)more negative aspects.

Table 2. Evaluation of proposed notations with SEQUAL framework

Notation	SEQUAL Framework [9] on Language Quality		
	Organizational Appropriateness	Domain Appropriateness	Comprehensibility Appropriateness
Traditional UML Act. Diag.	+	+	++
Annotated	+	+	++
Colours	++	++	++
Patterns	++	++	++

(++) more positive, (+) positive, (-) negative, (--)more negative aspects.

The two alternative notations are shown in Fig 1 and Fig 2, both capturing part of the work process within a home care unit, offering practical help and home nursing care to its clients. In the 'Mobile Care'-project, it is planned to better support the mobile aspects of the home care service by providing the employees continuous access to the central health information system (using a PDA to log/receive info) and other relevant systems from wherever they are using a combined PC/PDA-solution. The case description we used as inspiration for the models came from the 'Wireless Trondheim' project [8], which is currently managing and extending a mobile broadband (WLAN) infrastructure for Trondheim. *The home care involves services being offered in the home of the customer (elderly or ill people), including practical help and home nursing care. In the 'Mobile Care'-project, it is planned to better support the mobile aspects of the home care service by providing the employees continuous access to Gerica, the central system for storing health information on the home care patients and other relevant systems. Access is envisioned to be through software used in a smartphone/PDA (to log/receive info).*

The alternate process notations presented in Fig. 1 and 2 reflect the tasks by home care assistants who visit patients according to the list given by shift leader. Normally, the patient only needs help with day-to-day activities (e.g., shopping, cleaning, taking

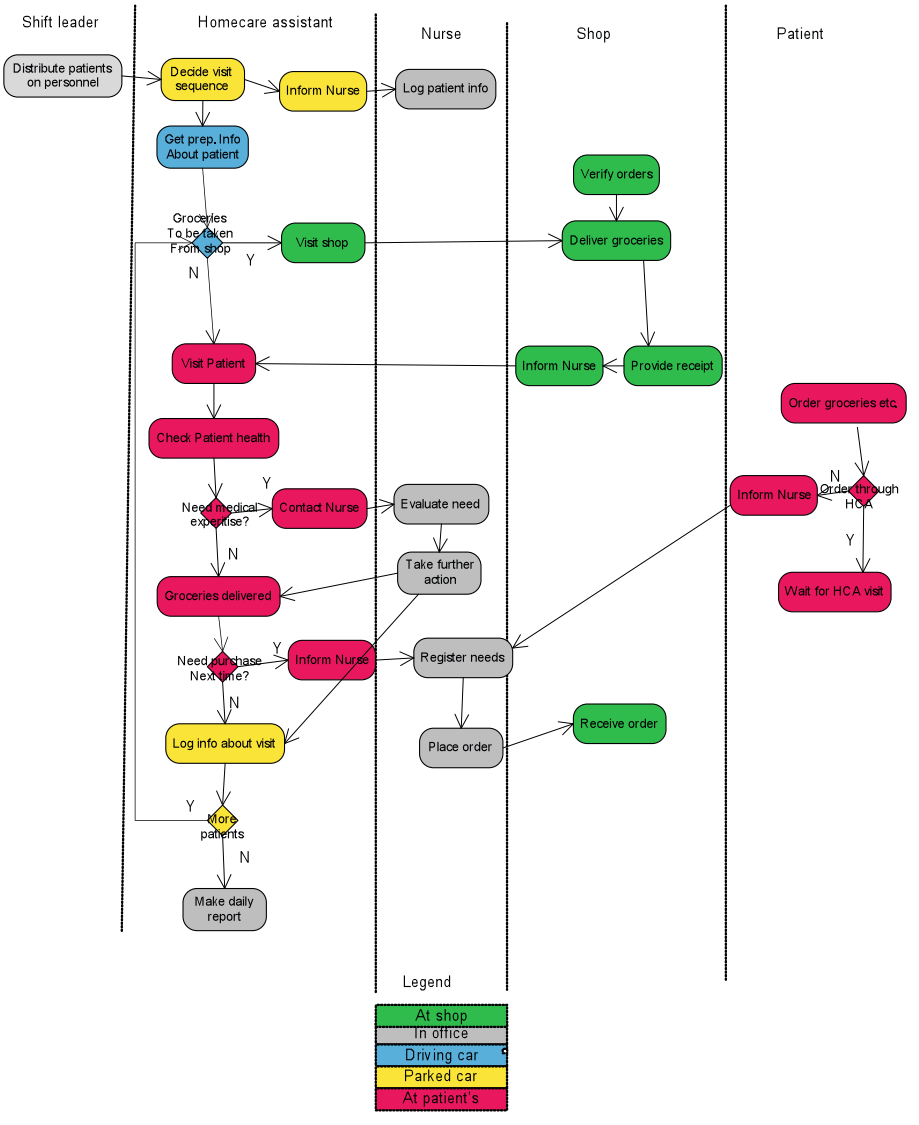


Fig. 1. Process example with colour fills

the right amount of medication), but in case there are some health complications that the assistant cannot handle alone, a nurse is contacted. Accessing Gerica from the PDA, the home care assistant logs the info about patients on the go informing patient health status. Also he picks up from shop the goods needed by the patient like toothbrush, lightbulbs and groceries, which are ordered through a nurse (so that he/she can make one bill). If the health care assistant needs further medical expertise he/she can request assistance from the nurse at the hospital through logging info in

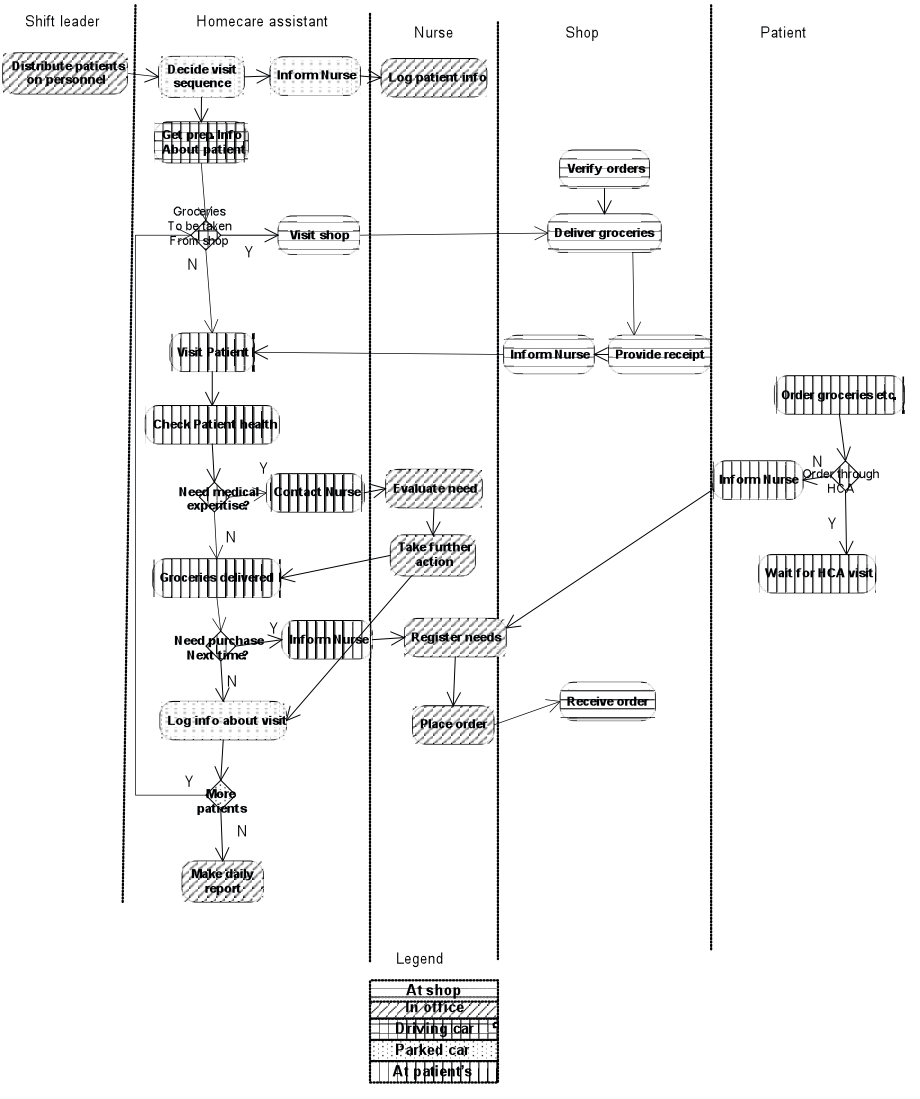


Fig. 2. Process example with pattern fills

Gerica. The nurses at the hospital get the request and provide further info/advice to be followed by the healthcare assistant (HCA). The health care assistant carefully note down (logs in Gerica) about the entire visit starting from office, on the car, at shop, at the patient's place until being back to the office. Finally the HCA finishes his job by reporting at the office about his/her whole day visit. Locations (e.g., office, car) and contexts (e.g., parked, driving) are indicated by colours and pattern fills.

For such experimental study on comparing modelling methods [21], it is important to show that the two notation alternatives and the cases are informationally equivalent. In other words, we do not want one language to be more comprehensive than the other. Since both notations are extensions of the same modelling language, adding the possibility to add location information in two different ways, we can argue for such equivalence.

Walderhaug et al. applied UML in the MPOWER project [5] with homecare services and claim that UML profiles [12] can be used as a mechanism for tool chains based on OMG's Model Driven Architecture (MDA) and UML standards [3] [9]. Work on mobile ontologies by Veijalainen [11] supports the idea of the 'where' aspect as essential in mobile processes, but excludes the 'what' aspect. Larsson [12] proposes the three building blocks for knowing the processes list How, What and Why, adds Who for use oriented design approach but omitted the 'Where' concept. Whereas the use of colour is not so common in conceptual modelling, in other areas of visual knowledge representation such as cartography colour is widely used to improve comprehension. Related to thisis the work of Bertin [13] on visual variables, where colour is one important differentiator. Moody [14], in his proposal for a 'physics' of notation based his work partly on the work of Bertin.

3 Research Method

Our goal is to compare the two notations presented above to see which one gives the best results for the user, in terms of understanding process diagrams and using such diagrams in problem solving activities. Such comparisons can be made in several ways, ranging from analytical comparisons, through controlled experiments on small-scale tasks, to usage in large-scale industrial projects. In this paper, our goal is to make a comparison of a colour-based and pattern-based notation, otherwise quite similar to a previous comparison of colour vs. annotations [7]. Hence it is natural to use a controlled experiment with similar structure as the previous one, and similar tasks: Having the participants answer true/false questions about a correct process diagram and thereafter having them identify defects in another diagram where we had deliberately inserted some mistakes versus the textual case description. Our only substantial change from the previous experiment - apart from replacing the annotated notation with a black/white pattern fill notation - was to make the process models somewhat more complex. This was done because of an observation in the previous experiment that some of the experimental tasks were slightly too simple, thus not distinguishing participant performance clearly enough. More complex models will also make the experiment slightly more realistic vs. the models used in industrial IS projects. In addition we elicited the participant's opinion about the notation through a post-task questionnaire. We thus had three main variables to measure about each notation in the experiment:

Understanding: the participant's fraction of correct answers to the true/false questions.
Error_detection: the participant's fraction of correctly detected errors to the total number of errors in the deficient diagram.

Average_opinion: the participant's average score on 14 questionnaire items about the notation.

Another question is whether to do a within-subjects (i.e., all subjects get both treatments) or between-subjects (half the subjects get one treatment, the other half the other treatment) experimental design. According to [15] a within-subjects design is advantageous if it is possible with the given treatments, since it doubles the sample size and also controls better for selection bias. Hence we chose this, as for the previous experiment, using a Latin Squares design with two different process model cases (one about home care as in Fig. 1, another about flight check-in). The Latin squares design is illustrated in Table 3.

Given the described variables, since the colour notation came out slightly better than the annotated notation in the analytical comparison in [6] and empirical comparison [7], we have the following key hypotheses with colour and pattern notation for our experiment:

H1: the Understanding scores for the colour notation (CN) will be better than those for the pattern notation (PN)
H2: the Error_detection scores for the colour notation (CN) will be better than those for the pattern notation (PN)
H3: the Average_opinion scores for the colour notation (CN) will be better than those for the pattern notation (PN)

Corresponding null hypotheses could also be formulated, but are not presented here for space concerns. Also, there could be more detailed hypotheses relating to different question groups investigated in the post-task survey, i.e., 5 questions related to Perceived Ease of Use, 5 to Perceived Usefulness, and 4 to Intention to Use. These hypotheses would be similar to H3, i.e., assuming that the colour notation would score better than the pattern notation. Again, these more detailed hypotheses are not shown to save space and found to be less significant.

57 students were recruited from a second year computer science class to take part in the experiment. With the Latin Squares design, these were divided into 4 groups according to which notation to try first, and on which case the notation was used, as shown in Table 3.

1. The 57 participants were randomly distributed to the four experiment groups in the Latin Squares design. The questionnaire prepared contained four parts, 1. Pre-experiment questionnaire, 2. Questionnaire on CN or PN with case 1 and post- experiment evaluation, 3. Questionnaire on PN or CN with case 2 and post-experiment evaluation. 4. Identifying mistakes in CN and PN with brief case explanation. The participants performed the following activities during the experiment: Answering a pre-experiment questionnaire: The purpose of the pre-experiment questionnaire was to investigate the participants' prior knowledge of related topics like UML, process modelling, etc., which can be used to control for any accidental group selection bias in spite of random selection (e.g., one group accidentally containing people with much more relevant experience). This is much less important for a Latin Squares design than for a between-subjects design, but since it only takes a couple of minutes for the participants to answer a few questions about their prior experience with relevant

modelling techniques, it still felt worthwhile to do. Questions investigated previous knowledge on modelling, UML, activity diagrams, specifications, IT work experience and knowledge about the domains the cases were taken from (home care and flight check-in), in total 8 questions that were to be answered within 5 minutes.

2. Reading a tutorial about the first diagram notation (pattern or colour), using a flight check-in case as an example case description followed by corresponding pattern or colour notation diagram were presented as a tutorial part in the experiment.

3. Being presented with experimental textual case description (home care or flight check-in), together with a diagram (pattern or colour), and at the end of this, participants must answer 12 true/false questions related to that particular case.

4. Answering a post-task questionnaire about the notation just used, containing 14 questions investigating Perceived Ease of Use (PEOU), Perceived Usefulness (PU), and Intention to Use (ITU) as inspired by the TAM model [16].

5.Repeating steps 2-4 using the other notation on a different case. Totally 54 minutes were allotted to complete steps 2-5 and return the booklet.

6. A separate booklet with textual description and notations on both cases deliberately seeded with some errors was distributed again and now the task was to find all the errors in the diagram (i.e., discrepancies between the diagram and the case description) i.e., all the students had questionnaire on both notations and both cases in steps 2-5 and also in 6. Of course, the seeded errors (5 errors per diagram) were the same both for the pattern and coloured variants of the diagrams. The allotted time to complete these tasks was 10 minutes. The case distribution is as shown in table 3.

Table 3. Latin square experiment questionnaire distribution to student groups

Group Id	**(Understanding+ TAM factor) Questionnaire on**	**Error Identification Questionnaire on**
Group A	Pattern Home Care + Colour Flight check-in	Pattern Flight check-in + Colour Home Care
Group B	Colour Flight check-in + Pattern Home Care	Colour Home Care + Pattern Flight check-in
Group C	Pattern Flight check-in + Colour Home Care	Pattern Home Care + Colour Flight check-in
Group D	Colour Home Care + Pattern Flight check-in	Colour Flight check-in + Pattern Home Care

4 Experiment Results

The results for the performance of the participants on the tasks of understanding (answering 12 true/false questions) and problem solving (detecting errors in diagrams relative to a natural language case description) are summarized in table 4. The means for understanding reflect the percentage of correct answers for each student, i.e., when using the colour diagrams students had on average 85% correct answers. For error detection it reflects the number of correctly identified model defects, i.e., on average 3.95 errors were found by those using the colour notation. As can be seen, the

performance on the true/false questions turned out with no practical difference between the treatments at all, while the error detection task had an effect of size 0.23 in favour of the colour notation. According to [17] this is only a small effect, and the difference was not significant, with a two-tailed p=0.09 for a paired t-test.

Table 4. Comparison of performances with the two notations

Compared variable (N=57)	Coloured diagram		Pattern fill diagram		Diffe-rence	Effect Size	Sign.? Y/N (p-value)
	Mean	SD	Mean	SD			
Understanding	0.85	1.26	0.86	1.56	0.01	0.002	No
Error detection	3.95	1.39	3.61	1.46	0.34	0.23	No

The results for the participants' opinions about the two notations, as indicated by their answers to the TAM-inspired post-task questionnaire, are shown in Table 5. The participants' opinion was strongly in favour of the colour notation, with effect sizes ranging from 1.15 to 1.60, which is a large effect according to [18]. As can be seen, these effects were strongly significant.

Table 5. Comparison of TAM factors with the two notations

Compared variable (N=57)	Coloured diagram		Pattern fill diagram		Diffe-rence	Effect Size	Significant? Y/N (p-value)
	Mean	SD	Mean	SD			
PEOU	3.13	1.77	1.97	0.72	1.16	1.15	Yes ($p<10^{-17}$)
PU	3.53	1.88	1.83	0.98	1.70	1.48	Yes ($p<10^{-17}$)
ITU	3.21	1.79	1.03	1.00	2.17	1.60	Yes ($p<10^{-17}$)
Average opinion	3.29	1.81	1.62	0.79	1.68	1.60	Yes ($p<10^{-17}$)

All in all, then, we have the following conclusions on our hypotheses:

- H1 was rejected, as we found no significant advantage for the coloured notation when it came to the measured understanding in terms of the scores for the true/false questions.
- H2 was also rejected; there was no significant advantage for the coloured notation when it came to measured problem solving capability in terms of the number of identified errors in the diagrams.
- H3 is confirmed due to the strong significant advantage for the colour notation when it came to responses to the post-task questionnaire investigating the participants' opinions about the notations and these data are documented in appendix.

Normally, one might think that a strong preference for one notation should also be accompanied by a strong advantage in performance, and vice versa. However, other experiments have also shown that this is not necessarily the case. In our previous experiment [7] the situation was opposite: The students performed better with the colour notation than the annotated one, but there was no significant difference in preference. Another similar example is [19], where one technique had a significant performance advantage, yet no such advantage was found for opinion, indeed there was no notable correlation between the participants' performance with a technique and opinion about that technique. So, a natural conclusion here seems to be that the participants have generally found the colour notation much more visually appealing than the pattern notation, but nevertheless they were still able to perform equally well with the black and white patterns.

5 Threats to Validity

Wohlin [20] suggests four relevant categories for discussing threats to validity in experiments: conclusion validity, construct validity, internal validity and external validity. *Conclusion validity* concerns the relationship between the treatment given and the outcome in measured variables. One important question is whether the sample size is big enough to justify the conclusions drawn, which can be investigated by means of the calculated effect size (ES). We accepted the hypothesis H3 of better overall preference for the colour notation (as well as sub-hypotheses for PEOU, PU, and ITU), with effect sizes as shown in Table 5.. Denoting the Type I error probability by α (accepting a relationship which really is not there) and the Type II error probability by β (overlooking a relationship that really was there), the following holds:

$$N = \frac{4(u_{\alpha/2} + u_{\beta})^2}{ES^2} \ .$$

If we use $\alpha = 0.05$ (our threshold for accepting a relationship as significant) and $\beta = 0.20$, we get $N = 32/(ES)^2$ [18] as a required sample size. Using the smallest effect size for opinion (PEOU), this means that we should have a sample size of at least 25 for the results that we have claimed as significant. We had 57 participants in our experiment, so we are clearly on the safe side here.

Construct validity is concerned with the inference from the measures made in the experiment to the theoretical constructs we were trying to observe (understanding, problem solving effectiveness). Of course, there are other ways to explore understanding than true/false questions after looking at a case description and diagram, and other ways to explore effectiveness than asking participants to identify errors. But at least, identification of errors is an important task in system development (for instance in connection with reviews / QA), and answering questions is at least one relevant way of testing understanding. Given the limited type (only true/false) and nature of the questions, it must still be admitted that they will not measure every aspect of understanding, and that more experiments with a wider range of experimental tasks would be necessary to draw more certain conclusions.

Internal validity means that the observed outcomes were due to the treatment, not to other factors. Our Latin-Squares experimental design was used to eliminate selection bias, and to control for any learning effects or effects of which case was used with which technique. In addition we performed a pre-experiment questionnaire to test whether other factors such as previous relevant experience could explain the differences between the various groups, not finding any such effects.

External validity is concerned with the question of whether it is possible to generalize from the experimental setting to other situations, most importantly to industrial systems development. The use of students instead of practitioners is a notable threat. However, this threat is reduced by the fact that we are only trying to compare two notations in relative terms, not evaluate their merits in more absolute terms. Moreover, these adapted notations would be new also to practitioners, thus reducing the advantage they might otherwise have had over students (e.g., if the practitioners had used the notations a lot at work).

6 Conclusion

For the development of mobile and multi-channel information systems it might be important to have process notations able to capture the "where" aspect, i.e., the location of the various activities. Hence, we have suggested some alternative adaptations of the UML Activity Diagram notation to enable the capturing of location. In this paper, we have reported on a controlled experiment comparing two such adapted notations, one using pattern and the other using colour to capture the location of activities. In a previous analytical evaluation, the colour alternative came out as slightly better, hence we hypothesized an advantage for the colour notation for all the variables we measured. Statistical analysis confirmed an advantage for the opinion, but not for understanding and error detection.

Even though we increased the diagram complexity versus our previous experiment [7] it must still be admitted that the experimental tasks are quite small and simple compared to what a professional IS analyst might be faced with at work. So, new experiments with even more challenging experiment tasks could also be of interest, as well as using modelling tools in the experiment instead of just pen and paper.

Finally, it would also be important to perform larger case studies in enterprise modelling / systems analysis projects, preferably in industry, to check whether advantages observed in a limited experimental setting also hold for real world usage. Such industrial evaluations would also give more insight on the workplace usefulness of a notation capturing location, e.g., in what cases would such a notation provide added value, and in what cases would it be better to keep on using a mainstream notation not capturing location.

References

[1] Dourish, P.: Re-Space-ing Place: “Place” and “Space” Ten Years On. In: Proc. ACM Conf. Computer-Supported Cooperative Work, CSCW 2006, Banff, Canada, pp. 299–308. ACM, New York (2006)

[2] Unified Modelling Language, `http://www.uml.org` (accessed 4.6.2010)

[3] Business Process Modelling Notation, http://www.bpmn.org/ (accessed 4.6.2010)
[4] Korherr, B., List, B.: Extending the UML 2 activity diagram with business process goals and performance measures and the mapping to BPEL. In: Roddick, J., Benjamins, V.R., Si-said Cherfi, S., Chiang, R., Claramunt, C., Elmasri, R.A., Grandi, F., Han, H., Hepp, M., Lytras, M.D., Mišić, V.B., Poels, G., Song, I.-Y., Trujillo, J., Vangenot, C. (eds.) ER Workshops 2006. LNCS, vol. 4231, pp. 7–18. Springer, Heidelberg (2006)
[5] Walderhaug, S., Stav, E., Marius Mikalsen, M.: Experiences from Model-Driven Development of Homecare Services: UML Profiles and Domain Models. LNCS (2009)
[6] Gopalakrishnan, S., Sindre, G.: Alternative Process Notations for Mobile Information Systems. In: Proc. I-ESA 2010, Coventry, UK (April 2010)
[7] Gopalakrishnan, S., Krogstie, J., Sindre, G.: Adapting UML activity diagrams for mobile work process modelling: Experimental comparison of two notation alternatives. In: van Bommel, P., Hoppenbrouwers, S., Overbeek, S., Proper, E., Barjis, J. (eds.) PoEM 2010. Lecture Notes in Business Information Processing, vol. 68, pp. 145–161. Springer, Heidelberg (2010)
[8] Andresen, S., Krogstie, J., Jelle, T.: Lab and Research Activities in Wireless Trondheim. In: Proceedings of IEEE International Symposium on Wireless Communication Systems, pp. 385–389. IEEE Computer Society, Los Alamitos (2007)
[9] Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modelling Language: User Guide. Addison-Wesley, Reading (1999)
[10] Lillehagen, F., Krogstie, J.: Active Knowledge Modelling of Enterprises. Springer, Heidelberg (2008)
[11] Veijalainen, J.: Developing Mobile Ontologies; who, why, where, and how? In: International Conference on Mobile Data Management, Manheim, Germany, pp. 398–401. IEEE, Los Alamitos (2007)
[12] Larsson, A.V.: Designing for use in a future Context – Five Case Studies in Retrospect, PhD thesis No: 1034, Institute of Tech., Linkoping Univ., Sweden (2003)
[13] Bertin, J.: Semiology of Graphics: Diagrams, Networks, Maps, University of Wisconsin Press (1983)
[14] Moody, D.L.: The "Physics" of Notations: Towards a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Transanctions on Software Eng. 35(6), 776–779 (2009)
[15] Field, A., Hole, G.: How to Design and Report Experiments. Sage Publications, London (2003)
[16] Davis, F.D.: Perceived usefulness, perceived ease of use and user acceptance of information technology. MIS Quarterly 13, 319–340 (1989)
[17] Zachman, J.A.: A framework for information systems architecture. IBM Systems Journal 26(3), 276–291 (1987)
[18] Hopkins, W.G.: A New View of Statistics. University of Queensland, Australia (2001)
[19] Opdahl, A.L., Sindre, G.: Experimental comparison of attack trees and misuse cases for security threat identification. Information and Software Technology 51(5), 916–932 (2009)
[20] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction. Kluwer Academic, Norwell (2000)
[21] Siau, K.: Informational and Computational Equivalence in Comparing Information Modeling Methods. Journal of Database Management 15(1), 73–86 (2004)

Appendix

In this section, some of the additional results, questionnaire information and detailed description of the other case (Flight check-in) of this student experiment are documented. This below section will give the reader a bird's eye view on how the experiment has been performed too.

A1 Flight Check in Case

Nowadays almost every airline provides online check in facility through PDA/laptop/other handheld devices to save time and as a cost effective measure. Consider a person (an air passenger) who is about to travel by flight that stay at a hotel. From the airlines he got a mail/sms regarding the possibility to make online check in notification before a reasonable time before his flight. As he is busy / unsure about his trip, he only decides to check in online a few hours before his flight, for example during his travel to the airport.

The airline system offers him different options, like a simple check in by replying SMS with YES, adding baggage within free limit and excess baggage for a fee, seat preference if he checks in before certain time limits. If it is not allowed to check in online, it also suggests whether it is possible to make airport check in or if check in is totally closed. In the case of that the check-in is closed, the system further suggests next available flights for free / for a fee. So the passenger can still make a travel by choosing the alternative flights with free of charge/fee suggested by check in/reservation system and proceed for online check in at one go or he can contact the customer care centre at the airport for further help.

As continuation of check in process, the air passenger adds baggage's and select seat for the flight. For confirming the check in process, if no fee is to be paid it confirms that the online check in is completed. Before confirmation, the system check for whether he needs to pay any fee and if it is it provides the option to pay online with credit / debit cards or pay at the check-in counter. On verifying that the payment is done, it confirms the check-in process complete. Fig. 3 is based on Color Notation (CN) and Fig. 4 Pattern Notation (PN) of a UML activity diagram reflecting the flight check-in case.

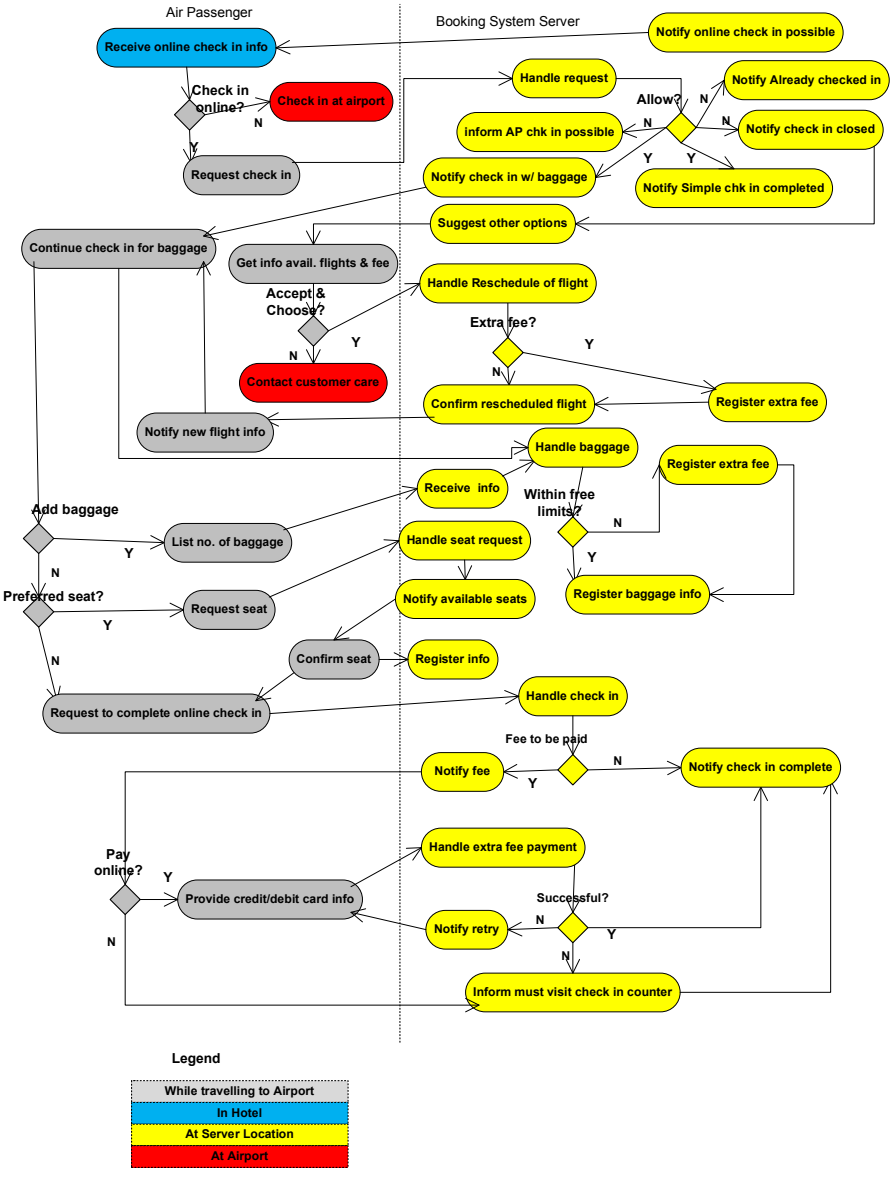


Fig. 3. Flight check in case- Color Notation (CN)

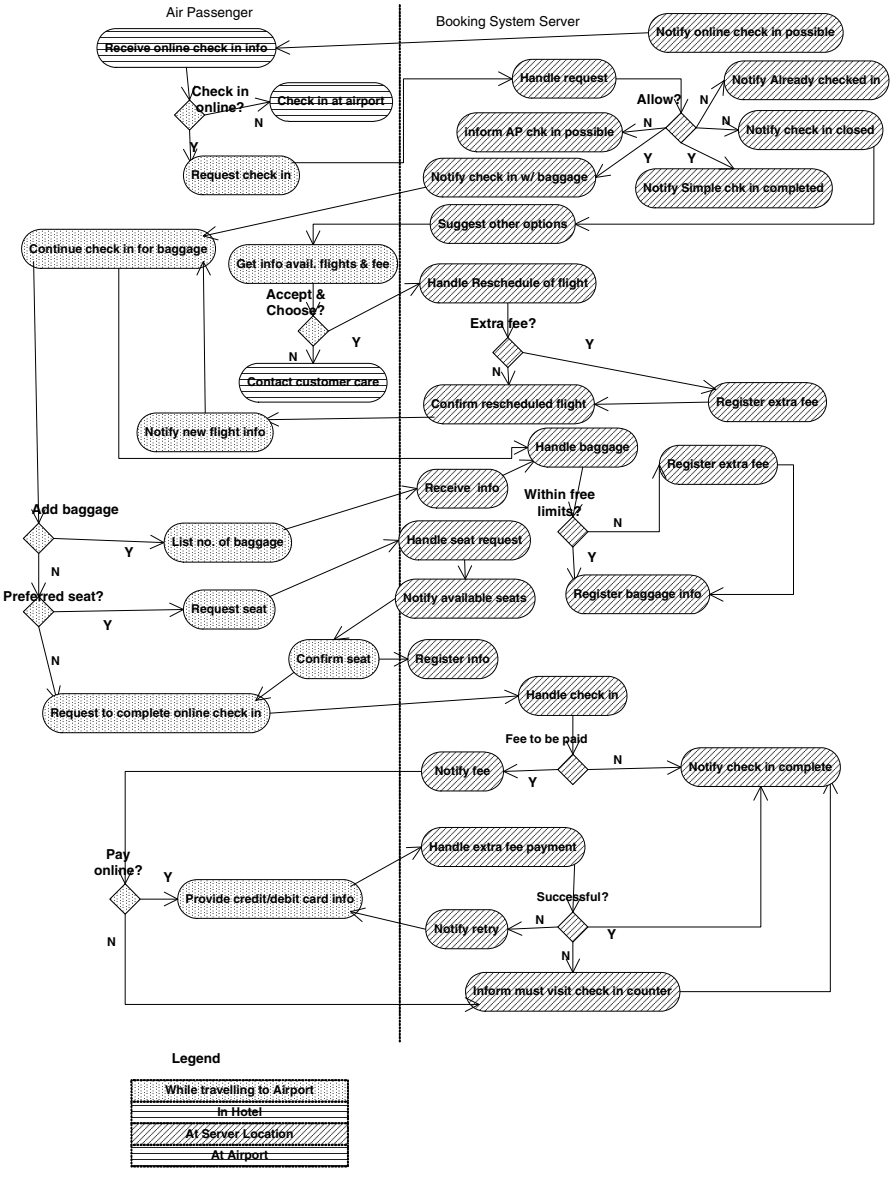


Fig. 4. Flight check in case –Pattern Notation (PN)

A2 Questions about Home Care Case

(Please write down whether the statements are True/False)

1 Home Care Assistant (HCA) receives the patient list to be visited at the Car.
2 HCA has been given the patient list along with visit sequence.
3 HCA contacts the nurse while driving the car usually.
4 If HCA needs medical experts suggestion HCA returns to office to meet Leader.
5 All the patients order groceries etc only through Nurse.
6 HCA collects all preparatory info about patients at the office itself.
7 Patients visited in sequence as per patient's desire.
8 HCA visits all the patients in the list at the patients place.
9 HCA delivers all the ordered groceries etc to the Nurse everyday.
10 Nurse comes down to patient's location and provide further info if it is needed by HCA.
11 HCA returns to office after visiting one patient in the list to log info about visit.
12 HCA starts and completes his/her duty at office.

A3 Questions about Flight Check in Case

(Please write down whether statements are True/False)

1. The passenger should only check in via online.
2. The passenger can add baggage's when check in online.
3. The passenger must choose a seat to complete check in process.
4. The passenger can reschedule his flight if he cannot check in right time.
5. If there is any fee on baggage he should pay online only.
6. For rescheduling the flight he must contact customer care centre at airport
7. Online check in process will not complete if the seat is not selected.
8. The passenger is allowed to check in at airport check in counter also.
9. The booking system will not allow online check in process until you get notification to do so.
10. In order to fly you must do airport check in if the system denies online check in because of your late online process.
11. If your credit card fails you can use another try to complete online payment if there is a fee on the course of inline check in.
12. The system always notifies if the online check in process completed.

A5 Post Experiment Questionnaire and Results

The TAM model [16] was used with the three factors Perceived ease of use (PEOU), Perceived usefulness (PU) and Intention to use (IU). As a part of this evaluation fourteen questions was presented for the students to answer based on their experience from using these two notations. On evaluating these experiences it is found that the colour notations outperformed significantly the pattern notation in all three PEOU, PU and IU aspects. Post-experiment questionnaire with 14 questions with results are Fig. 5 and in table 6.

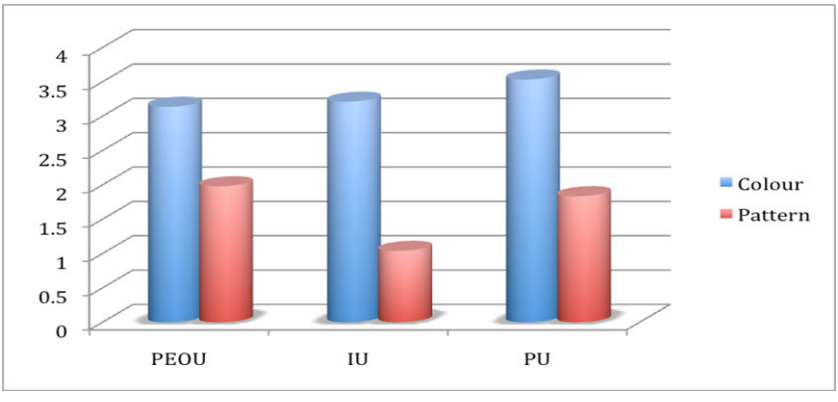


Fig. 5. Post-experiment TAM Questionnaire analysis

Table 6. Results with TAM questionnaire on two notations

Question id.	TAM Questionnaire	Related TAM factor	Colour Notation (max.avg. value :4)	Pattern Notation (max.avg. value :4)
1	Notation gave me a better understanding of the activity where it is performed	PEOU	3.754	2.263
2	I found this notation is very easy to master	PEOU	3.772	1.965
3	I would have found where the activities has been performed by using common sense.	PEOU	1.404	1.438
4	I found very easy to use and recognize this notation	PEOU	3.596	1.614
5	I was not often confused about how to apply this notation to UML activity diagram	PEOU	3.158	2.614
6	If I need to identify where the activity process done in a future project, I would use this Notation.	IU	3.193	0.965
7	I will try this notation if I been assigned in my future work involving mobile process	IU	3.105	0.982
8	If I am working as freelance consultant for a customer who needs help finding where the activities (mobile processes) are performed in his system, I would use this notation in discussions with that customer.	IU	3.211	0.965
9	If I am employed in a company which discusses what technique to introduce for modelling mobile IS and someone suggest this notation, I would support that.	IU	3.333	1.246
10	The notation made the activity diagrams more systematic.	PU	3.421	1.719
11	It is very easy to get used to the Notation in a project	PU	3.526	1.947
12	I can read and understand this notation quickly.	PU	3.719	1.333
13	This notation is easy to remember	PU	3.544	2.211
14	This notation made me more productive in finding where the activities been performed.	PU	3.456	1.965

vBPMN: Event-Aware Workflow Variants by Weaving BPMN2 and Business Rules

Markus Döhring and Birgit Zimmermann

SAP Research
Darmstadt, Germany
{markus.doehring,birgit.zimmermann}@sap.com

Abstract. When workflows are modeled for practical use, workflow variants often have to be considered to fit dynamically changing context factors. If there is a rich workflow context with a large value space, contemporary BPM solutions lack the support for on-the-fly generated variants, requiring explicit one-by-one modeling instead. Researchers have recognized the value of business rules for variant and adaptation support. However, there is still a need for dedicated standards-based constructs for context-dependent event- and exception-handling. Motivated by a realistic example, we therefore foster a framework for the combined use of business rules with a BPMN adaptation pattern catalogue. As the core contribution of this work, we substantiate our framework with a metamodel called vBPMN, which is weaved from BPMN2 and the R2ML rule language and allows for the convenient definition of variant models.

Keywords: process variants, process adaptation, flexible workflows, business rules, context-awareness, BPMN.

1 Introduction

The practical necessity for supporting the modeling and execution of large amounts of workflow variants has recently been addressed in research [1,2]. Challenges relate to modeling complexity on type level at design-time as well as variant creation at runtime by instance adaptation. To tackle these challenges in an integrated manner, we propose a combination of workflow-, rule- and event-modeling [3]. However, especially event-awareness, i.e. the ability to differently react on external status changes in different data-contexts, has not yet been sufficiently integrated into approaches for handling workflow variants. Moreover, existing approaches are rarely aligned with industry standards and lack guidance for systematic and incremental management of variants.

As a solution, this paper promotes a framework for supporting BPMN2-based event-aware workflow variants. We have presented the corresponding basic ideas in [4,5]. Adding upon our works, we provide a more comprehensive realistic example workflow and explicitly state the derived requirements for our approach in Section 2. The enhanced framework itself is presented in Section 3. As the core novel contribution of this paper, in Section 4 we substantiate our approach with

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 332–341, 2011.

a metamodel which weaves and extends BPMN2 and the R2ML rule language for pattern-based variant modeling. Section 5 discusses related approaches and Section 6 concludes this work.

2 Requirements for Event-Aware Variant Support

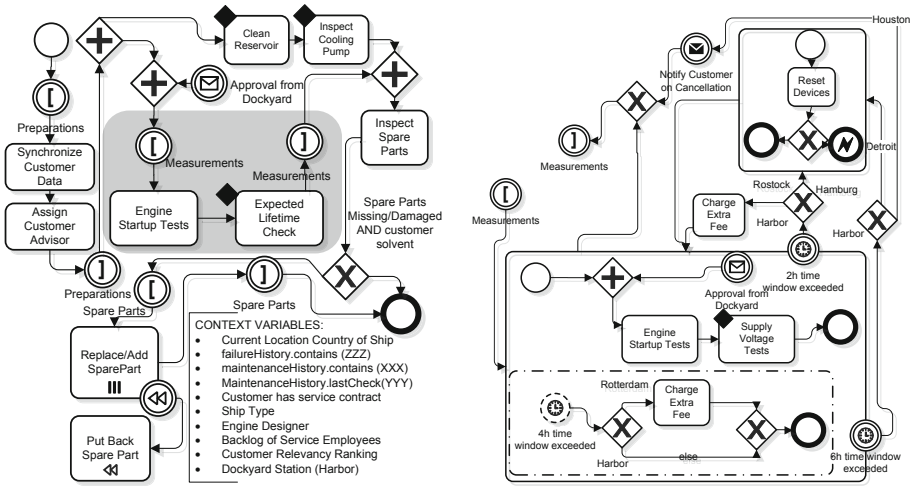
2.1 Motivating Example: Maintenance Service Workflow

To motivate the need for event-aware workflow variants, we first introduce a fictitious but realistic example workflow from the domain of ship engine maintenance service provision. It will also be used as a running example in this paper. Figure 1(a) shows a workflow for service execution in BPMN 2.0 notation[1]. BPMN2 was chosen due to its status as a de-facto industry standard and its facilities for execution details specification as well as an XML serialization format. The intermediate events with square brackets as well as the black diamonds on the tasks in Figure 1(a) are explained later in Section 3.

In the upper part of the parallel branch of the workflow, tasks for maintaining the ship's cooling system are executed. In the lower part, engine startup tests and a subsequent lifetime analysis of the motor are conducted. For these tests, the engineers on the ship have to wait for an approval, since only few ships may run their engine in the harbor at the same time for environmental reasons. Then, spare parts which are permanently stored on the ships are inspected and eventually replaced, but only if the customer is solvent w.r.t. a positive credit report. If the service provision is canceled in some rare cases, it might be necessary to revoke even already replaced spare parts from the ship. For illustration purposes, Figure 1(a) also contains a list of context variables for service execution workflow instances. Some of them may dynamically change while instance execution, like the backlog of service engineers.

In many situations, a workflow instance needs to be tailored to its context, abstracted by the values of its context variables. This introduces additional complexity at design- and runtime, especially if external status changes (events) should be taken into account and immediately be reacted upon in a context-specific manner. In the *measurements* part (marked gray) of the workflow, it might for instance be the case that the service engineers do not dispose of an arbitrarily long time frame for conducting the tests. The actual time frame and the reaction when it is exceeded depends on the context variable *dockyard station*. Traditional workflow systems would require the explicit modeling of a variant per harbor, resulting in an unmaintainable amount of variants containing partly redundant business logic. Another option would be modeling everything into one aggregated model. Just for the measurements segment with 6 different harbors, this would look like in Figure 1(b). For instance, if a time window is exceeded in Houston, the tests are canceled and a notification to the customer is given. In Rostock and Rotterdam, an additional service fee is charged to the customer, but the tests only have to be canceled and repeated in Rostock.

[1] http://www.omg.org/spec/BPMN/2.0/PDF/



(a) Workflow for Ship Engine Maintenance Service Execution

(b) Hardly Maintainable Workflow Part *Measurements* (Including all Variants)

Fig. 1. Example for Context-Dependent Ship Engine Maintenance Workflow

It is already difficult to tell from the graphical model which ones are reference workflow parts and which ones are contextual facets. If even additional context variables for event handling would be included, the workflow graph would literally blast and become even more unmaintainable.

2.2 Requirements for Variant Support Framework and Metamodel

Based on examples as above from industry research in logistics, retail and field services in the project *Allianz Digitaler Warenfluss* (www.adiwa.net) and previous related work (references given below), we define the following requirements for a framework and metamodel supporting flexible variant modeling:

- For "factorizing" the modeling complexity, variants should be realized by an efficient combination of workflow models and business rules [1,6]. As shown, purely workflow based models explode in complexity when the data context is too large, while according to [7] purely declarative approaches are sparsely understood and tooling is not yet mature enough. Methodologies for *transforming* declarative business logic into imperative languages like BPMN exist [8]. However we still need a dedicated meta-model which supports the *integration* of declarative and imperative components for variant modeling.
- As the modeler should not be required to reinvent flexibility constructs for every modeling project, such efforts have to be be guided in a pattern-based manner [5]. Therefore, adding to the state-of-the art, our metamodel must especially facilitate the definition of reusable and nestable adaptation patterns also for context-dependent event- and exception handling.

– To ensure exchangeability of existing workflow and rule definitions [9], it is desirable to have a platform-independent metamodel which builds upon established standards from industry or research. Those standard meta-models should either be directly executable in existing engines, or there should be transformation mechanisms available which are able to generate platform-specific syntax from the respective metamodel parts.

In the following, we address these requirements and define how the presented concepts add up to a comprehensive framework for event-aware variant support.

3 A Framework for Event-Aware Workflow Variants

3.1 Adaptive Workflow Segments

In our framework, we build upon the general approach of [1] for specifying a reference workflow with a data-context and adaptive segments which can be subject to context-dependent structural adaptations. We restrict an adaptive segment to be any block-structured partial graph of the (eventually graph-structured) workflow, i.e. having only a single entry and a single exit point. Even if adaptations may be non-local in terms of concerning multiple adaptation segments, this facilitates the modular definition of nestable adaptation patterns in Section 3.2 and the checking of their consistent joint use. Figure 1(a) contains two different conventions for annotation. Adaptive segments can be marked with enclosing intermediate throw event nodes carrying opening or closing square brackets ⓘ ⓘ. Or, implying same semantics but being more space-saving, a black diamond ◆ can be put in the upper left corner of a single task. The choice of explicitly modeling the adaptive segments by enclosing intermediate events enables the workflow engine to get notified whenever an adaptive segment is entered or left. Figure 1(a) further shows how the data context is modeled. The execution semantics for adaptive workflow segments are informally defined as follows: If an entry to an adaptive segment is signaled for a workflow instance, the context variables are evaluated and the segment eventually becomes subject to immediate adaptations before continuing through the segment. At each entry time to an adaptive segment, a consistent isolated variant segment is created which does *not* change even though a variable may change while the variant segment is executed. If a variable changes while segment variant execution, the change is either ignored or more sophisticated checking and error resolution mechanisms have to be considered [1]. However, if the same adaptive segment is entered multiple times, for example via a cycle, the execution semantics allow for the creation of multiple different variants of the same segment.

3.2 Event-Aware Adaptation Patterns

The structural adaptations which can take place when entering an adaptive segment are defined in a BPMN2 adaptation pattern catalogue which has been presented in [5]. It contains patterns for realizing basic behavioral deviations

from the reference workflow like task insertion and skipping as well as more sophisticated patterns for immediate context-dependent reactive behavior on events as motivated in Section 2. Each pattern consists of an implicit parameter *<AdaptationSegment>* relating to which workflow segment it is applied on. For some patterns, additional parameters relating to model elements from the weaved metamodel defined in Section 4 are specified.

In Figure 2, one example exception handling pattern of our catalogue is presented. It realizes the generic treatment of timeouts with a handler task and eventual subsequent escalation or restart, concretely for the harbors Detroit and Hamburg in Figure 1(b). It is shown in Figure 2 how the pattern is weaved with the adaptation segment at runtime, before a token continues after passing the segment entry. With our framework, it is consequently possible to invoke this type of exception handling only for specific situations as variants and to conveniently combine it with other adaptation patterns like task insertions. We consider this feature as a key potential for efficient variant modeling and maintenance. The use of more sophisticated patterns like context-based synchronization of workflow branches entail some consistency constraints, as such adaptation patterns are for example problematic to be used within loops. More elaborations on this topic can be found in the paper describing our pattern catalogue [5]. We are currently working on a comprehensive description and formalization of such constraints. One main advantage of our framework compared to other similar approaches is the formalization of the patterns themselves as parameterizable BPMN2 fragments. The main benefits are a more intuitive understanding of the adaptations by the modeler who is already familiar with BPMN, a better starting position for later consistency checks of the overall variant model and especially a better modifiability and extensibility of the patterns themselves.

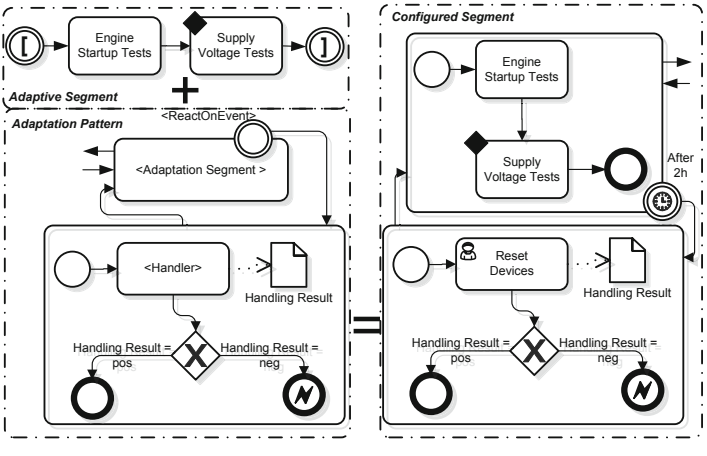


Fig. 2. Example of an Adaptation Pattern and its Application to an Adaptive Segment

3.3 Rule-Based Application of Patterns at Runtime

The connection between data-contexts and adaptation pattern applications can be established by formulating rules in an event-condition-action (ECA) format. The event of such an ECA rule corresponds to the entry event of an adaptive segment. The conditions constitute value restrictions on context variables. Finally, the actions contain parameterized adaptation patterns from the catalogue. They are at least parametrized with the adaptive segment belonging to the entry node of the rule's triggering event. We can define a pseudo-syntax as follows:

ON *entry-event* **IF** *<data-context>* **THEN APPLY** *<pattern(segment, parameters)>*

Below, some example rules for the example workflow in Figure 1(a) are given, partly in a hierarchical manner for better navigability and maintainability by the modeler. Rules #1-#3 skip or add particular tasks for example depending on whether there are special occurrences in the maintenance history of the engine. Rule #4 realizes the *timeout* contextual facet for the harbor in *Hamburg* presented in our previous example in Figure 1(b). Some execution problems when applying adaptation rules are conveniently prevented by our approach. Since the adaptations are achieved by parametrized modular patterns and each pattern gets at least the segment block as a parameter, it is for instance not a problem that rules #2.1 and #3 can be applied at the same time without crashing the engine due to missing references. Here, the execution order of patterns might be of relevance. We aim to tackle this in future by introducing constraints on joint pattern usage or a pattern application ordering.

```
RULE #1: ON expectedLifetimeCheck_entry IF existingServiceContract AND currentBacklog<80\%
 THEN APPLY insert_parallel(segment='expectedLifetimeCheck_entry', task='systemCalibration')
RULE #2: IF !(existingServiceContract) THEN -
  RULE #2.1: ON inspectCoolingPump_entry IF -  THEN APPLY delete(segment='inspectCoolingPump')
  RULE #2.2: ON spareParts_entry IF -  THEN APPLY delete(segment='spareParts')
RULE #3: ON inspectCoolingPump_entry IF yearsFromStartup(pumps)>4 OR yearsFromRepair(pumps)>8
 THEN APPLY replace(segment='inspectCoolingPump', task='replaceCoolingPump')
RULE #4: ON measurements_entry IF dockyardStation='Hamburg'
 THEN APPLY timeResolveRepeat(segment='measurements', HandlerTask='resetDevices', timer=2h)'
```

4 Weaving BPMN2 and R2ML into vBPMN

As stated in Section 2, our metamodel should reuse existing industry standards or promising metamodels from state-of-the-art research where useful and reasonable. For the definition of event-aware workflow logic, we have already motivated the employment of BPMN2. Moreover, its specification includes extensibility mechanisms which can directly be used non-invasively to integrate BPMN2 with our metamodel, however corresponding details are omitted in this paper due to space restrictions. For the specification of rules, we require a language which offers a rich syntax for expressing context constraints and event-triggered parameterizable actions and which also provides a meta-object facility (MOF) metamodel together with an XML-based exchange format. Therefore we selected

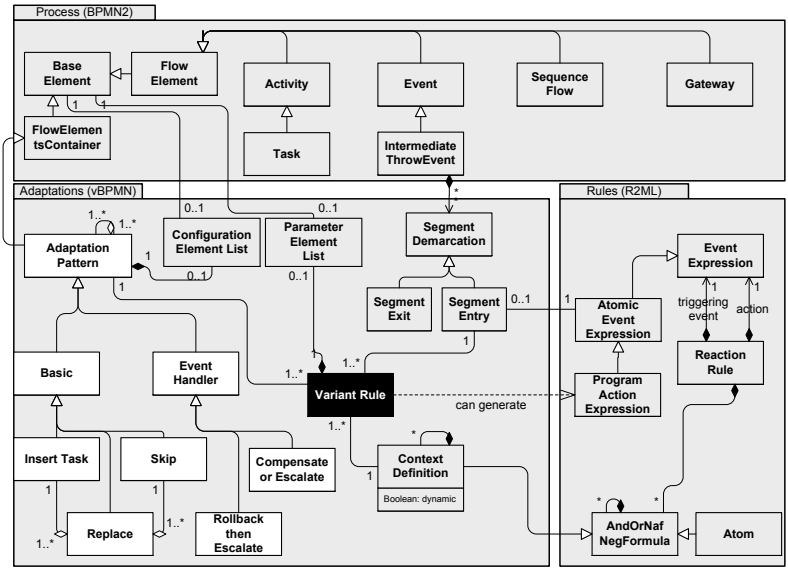


Fig. 3. vBPMN Metamodel Package Weaved with BPMN2 and R2ML

R2ML[2], a rule markup language which integrates OCL, SWRL and RuleML, includes four rule categories (derivation, production, integrity and reaction rules) and allows for transformation-based rule interchange between main rule engines like, e.g., Jess or jBoss Drools. As Figure 3 illustrates, our concept for adaptation-pattern based variant definition can be defined in a metamodel weaved and extended from BPMN2 and R2ML called *variant BPMN* (vBPMN).

The required elements from BPMN2 and R2ML are imported into the vBPMN package, where they are enriched and associated to allow for a convenient representation of the variants. *AdaptationPatterns* as presented in Section 3.2 extend the BPMN2 *FlowElementContainer*. As such, patterns can be defined using regular BPMN2 flow elements, which however may be associated to a list of parameters (*ConfigurationElementList*) which are provided only at configuration time. The patterns may be defined in a recursive manner, i.e. a pattern like *replace* might be composed by reusing other patterns like *insert* and *skip*. *SegmentEntry* events trigger R2ML *ReactionRules*, which are associated to a vBPMN data *ContextDefinition* via an R2ML *AndOrNafNegFormula*. Note that eatures like crosscutting concerns of workflows [10] can be conveniently realized by re-using *AtomicEventExpression* for multiple segment entries in the same or in different workflows. A *ProgramActionExpression* (= rule action) which determines a parametrized application of an adaptation is not explicitly maintained in the model, as it can be automatically derived from the information attached to a *VariantRule*. A significant advantage of this weaved metamodel compared to other proprietary approaches for workflow adaptation or variant handling is,

[2] https://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/6 (v0.5)

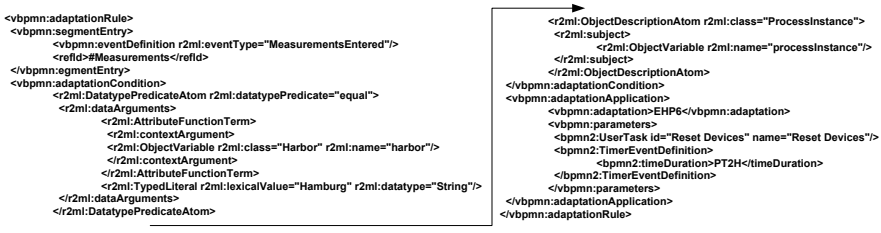


Fig. 4. Example XML Adaptation Rule and Pattern-Based Configuration

that the rule and workflow parts may also exist in isolation without our vBMPN extension. That means, one can for example generally define and maintain the reference workflow, adaptation patterns and context constraints in a favored IDE and import or export them in resp. from our framework. To be able to profit from established model-driven-engineering features of OMG's MOF like for instance OCL for model constraints and to provide a technical foundation, e.g., for the future development of a GMF-based graphical editor, we implemented the vBPMN metamodel in Ecore. The existing BPMN2[3] and R2ML[4] Ecore files were taken as basis. Figure 4 illustrates the already discussed variant rule #4 in an XML format based on the vBPMN Ecore metamodel. It represents the example in Figure 2, showing how the variant rule parametrizes the exception handling adaptation pattern and applies it on the workflow's adaptive segment *Measurements* at runtime. At this point it is implied that any task which is put in for <*Handler*> is able to deliver a compatible data structure <*HandlingResult*> which can be used for evaluating the gateway conditions. It is strongly recommended to dispose of such design-time validation mechanisms for dynamically adapted workflows also on the data-flow level. Generally, we are currently working on a comprehensive catalogue of OCL constraints for our weaved metamodel. Examples concern the enforcement of proper pairwise usage of adaptive segment entry and exit nodes only on block-structured parts of the workflow, or the restricted use of special synchronization patterns as explained before. For the ability to execute our variant model, we extended jBoss Drools 5.1 for interpreting variant rules at runtime and presented a prototype in [4].

5 Related Work

Prominent approaches like [2] propose the modeling of one large reference workflow, where parts of it may be faded out at runtime based on data constraints. For some scenarios however, the initial modeling of all such variants may be infeasible and a stepwise and extensible rule-based approach may be required. The framework presented in [11] reflects contextual changes by "migration arcs" between distinct process fragments which can be used within a context-sensitive

[3] http://git.eclipse.org/c/bpmn2/tree/org.eclipse.bpmn2/model
[4] http://www.emn.fr/z-info/atlanmod/index.php/Atlantic\#R2ML_4.0

workflow region. However, these fragments also have to be modeled one-by-one and can not be composed in a convenient way. Comparable to our vBPMN metamodel, the authors of [9] extend the BPMN metamodel to increase workflow flexibility at runtime. BPMN and R2ML are weaved as rBPMN to form model elements of higher expressiveness, like a rule-based gateway. Those can however not be conveniently employed for variant management as in our framework. A potential merging of rBPMM and vBPMN could be carefully considered, since the approaches and metamodels are complementary. In [8], it is shown how a BPMN model can be derived starting from a declarative fact-based data model. Although containing promising ideas, the approach does not address the challenge of declaratively *combining* process aspects based on workflow standard notation. Similar to our approach, [12] and [10] foster the modularized definition and dynamic reuse of process fragments. The latter authors moreover motivate the need for modularization by cross-cutting concerns in BPMN and discuss the extension of workflows with aspects in an AO4BPMN syntax. In our work, cross-cutting concerns can easily be realized by re-using adaptation segment nodes of the same type multiple times in a workflow. However, we generalize the usage of parameterizable BPMN fragments to establish efficient variant management also allowing the realization of context-dependent event-awareness.

The authors of [1] focus on the configuration and management of workflows with the help of adaptation rules. The work in [13] contains a sophisticated constraint network which allows the validation of variants resulting from ad-hoc task execution at runtime. However, these frameworks do not explicitly consider eventing features in terms of immediate context-dependent reactions on external status changes or the provision of a supporting corresponding pattern catalogue.

6 Conclusion and Future Work

In this work, we have motivated the need for conceptual guidance for workflow variant support based on a comprehensive practical example, which showed that workflow diagrams explode in complexity if not sufficiently integrating declarative approaches. Existing approaches and their meta-models typically focus on isolated features like modularization or adaptation, but especially lack support for realizing variant behavior w.r.t. eventing concepts and neglect the integrated consideration of declarative and imperative, i.e. rule and workflow modeling approaches building upon established standards. We already extended state-of-the-art for rule-based variant configuration with a catalogue of basic- as well as event-aware adaptations formalized in BPMN2 which can be applied on adaptive segments in a reference workflows for variant configuration at runtime.

As a main contribution of this work, we presented vBPMN, a meta-model weaved from BPMN2 and R2ML for expressing rule-based adaptations using BPMN as a facility to specify structural adaptation patterns and combining them with the declarative power of defining events and context-conditions in R2ML. The metamodel has been implemented in Ecore.

We are currently enhancing vBPMN by targeting a reasonable set of OCL constraints for the general vBPMN metamodel as well as for the concrete

patterns in our catalogue to provide means for ensuring design-time consistency of the variants. As future work, the enhancements will also include the consideration of data-flow issues when dealing with adaptations at runtime. Regarding the applicability of our approach, we plan to evaluate the modeling of variants by rule- and pattern-supported workflow adaptations for a marine service scenario of a globally operating engine manufacturer within our research project.

Acknowledgements

This work was developed in the project *Allianz Digitaler Warenfluss* (ADiWa) that is funded by the German Federal Ministry of Education and Research. Support code: 01IA08006.

References

1. Hallerbach, A., Bauer, T., Reichert, M.: Configuration and management of process variants. In: Intl. Handbook on BPM, pp. 237–255. Springer, Heidelberg (2010)
2. La Rosa, M., Dumas, M., ter Hofstede, A.H.M., Mendling, J.: Configurable Multi-Perspective Business Process Models. IS 36(2) (2011)
3. Döhring, M., Karg, L., Godehardt, E., Zimmermann, B.: The Convergence of Workflows, Business Rules and Complex Events. In: ICEIS 2010, Funchal, pp. 338–343 (2010)
4. Döhring, M., Zimmermann, B., Godehardt, E.: Extended Workflow Flexibility using Rule-Based Adaptation Patterns with Eventing Semantics. In: Informatik2010 Service Science, Leipzig. LNI, pp. 216–226. GI, Bonn (2010)
5. Döhring, M., Zimmermann, B., Karg, L.: Flexible Workflow at Design- and Runtime using BPMN2 Adaptation Patterns. In: BIS 2011, Poznan. Springer, Heidelberg (accepted 2011)
6. van Eijndhoven, T., Iacob, M.-E., Ponisio, M.L.: Achieving Business Process Flexibility with Business Rules. In: EDOC 2008, Munich, pp. 95–104. IEEE, Los Alamitos (2008)
7. Weber, B., Reijers, H.A., Zugal, S., Wild, W.: The declarative approach to business process execution: An empirical test. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 470–485. Springer, Heidelberg (2009)
8. Bollen, P.: BPMN as a Communication Language for the Process- and Event-Oriented Perspectives in Fact-Oriented Conceptual Models. In: OTM 2009 Workshops, pp. 639–648. Springer, Heidelberg (2009)
9. Milanovic, M., Gasevic, D.: Towards a Language for Rule-Enhanced Business Process Modeling. In: EDOC 2009, Auckland, pp. 64–73. IEEE, Los Alamitos (2009)
10. Charfi, A., Müller, H., Mezini, M.: Aspect-oriented business process modeling with AO4BPMN. In: Kühne, T., Selic, B., Gervais, M.-P., Terrier, F. (eds.) ECMFA 2010. LNCS, vol. 6138, pp. 48–61. Springer, Heidelberg (2010)
11. Modafferi, S., Benatallah, B., Casati, F., Pernici, B.: A Methodology for Designing and Managing Context-Aware Workflows. In: MOBIS 2005. Springer, Heidelberg (2005)
12. Ma, Z., Leymann, F.: BPEL Fragments for Modularized Reuse in Modeling BPEL Processes. In: ICNS 2009, Valencia, pp. 63–68. IEEE, Los Alamitos (2009)
13. Lu, R., Sadiq, S., Governatori, G.: On managing business processes variants. Data & Knowledge Engineering 68(7), 642–664 (2009)

Analyzing the Integration between Requirements and Models in Model Driven Development

Iyad Zikra, Janis Stirna, and Jelena Zdravkovic

Department of Computer and Systems Sciences, Stockholm University
Forum 100, SE-164 40 Kista, Sweden
{iyad,js,jelenaz}@dsv.su.se

Abstract. In Model Driven Development (MDD), models replace software code as the development artifact. At the same time, requirements represent the information that is elaborated in models. However, despite the tight relationship between models and requirements, only a few MDD approaches provide the necessary methodological guidelines and tool support to explicitly facilitate this relationship. We analyze approaches for integrating requirements with models within MDD and highlight the common characteristics, benefits, and problems. Based on the analysis, we elicit a set of general properties that need to be fulfilled when considering the integration of requirements and models, and we assess the contribution of the considered approaches accordingly.

Keywords: Model Driven Development, MDD, Model-Driven Engineering, MDE, Requirements.

1 Introduction

Model Driven Development (MDD) is an approach to software development whereby expert knowledge for creating models of the desired system is formalized, enabling automatic transformation of the models into an executable form [37]. Despite being used for a long time, models have been viewed as secondary artifacts, created for the purpose of documenting and communicating information system requirements and designs. In reality, models are often abandoned as soon as they have served their immediate purpose [37]. MDD proposes the use of models as the essence of software production, elevating them to primary artifacts that need to be well designed, maintained, and delivered as part of the final product [19]. In essence, MDD strives towards increasing the abstraction level of development by replacing software code as a development artifact with conceptual models.

At the same time, collecting the correct requirements has been widely recognized to be a major factor for the success of software development. Requirements represent the information that is elaborated in models during later development stages. Hence, there is a tight relationship between requirements and the models which specify how the requirements are met. This is often overlooked by MDD approaches and tools. While MDD requires transformations to be clearly defined between different models, the initial model in the MDD process is left for the analyst to create from a separate requirements specification (Fig. 1). As a result, the benefits of using MDD to improve

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 342–356, 2011.

efficiency of software development are lost at the start of the process. Capturing the relationship between requirements and MDD models can:

- enhance the understanding of the rationale behind system design decisions;
- facilitate the propagation of changes from requirements to models;
- improve the correctness and completeness of models and the final software;
- increase the potential of reuse for models that stem from similar requirements.

The gap which exists between requirements and models in MDD is noticeable. Despite some attempts to establish a connection (see Section 3), the problem is further emphasized by limited tool support and insufficient research addressing it [3].

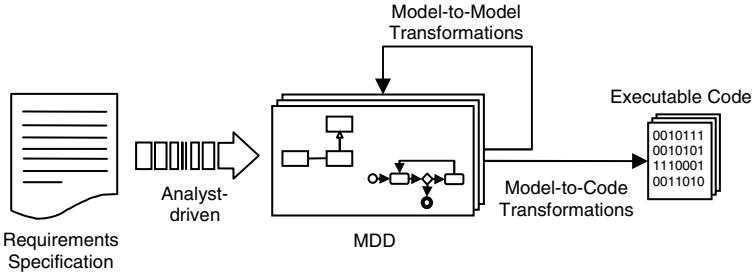


Fig. 1. Overview of the MDD process

The objectives of this paper are (1) to survey existing contributions to integrating requirements in MDD and (2) to elicit and formulate a set of properties for such integration based on the results of the survey.

The remainder of the paper is organized as follows: Section 2 gives an overview of MDD. Section 3 includes a discussion of the surveyed approaches for integrating requirements and models. The properties of integration are presented in Section 4, and the concluding remarks are given in the final section.

2 Model Driven Development

Until recently, advanced programming languages, like Java or C#, have been sufficient for developing software systems. The details of complex systems have been managed using middleware platforms, such as J2EE or .NET, and supported by programming-like notations, like the XML-based languages. However, the increasing complexities of the tasks which software needs to perform, and the growing demand for shorter development cycles, have exposed the limits of technologies that are based on coding [52]. Models are recognized as suitable candidates for increasing the abstraction level of development and for supporting the migration through different phases of development (analysis, design, implementation, etc.). They can be used to increase the quality, effectiveness, and reliability of software development [52]. To exploit the full potential of automation, complete and functional programs should be generated from models. Additionally, the models need to be automatically verified to ensure they correctly describe the desired information system.

MDD [21] refers to those emergent and model-centric development approaches, where models are essential parts of the final product. Models in MDD steer the development process and carry vital information in various forms. MDD enables developers to create abstractions of what needs to be built, or the problem domain, as opposed to abstractions of how it will be done, referring to high-level development platforms and languages, called the solution domain [51]. Those abstractions, or models, help in filling the problem-implementation gap that separates the problem domain from the software solution [19].

MDD is also concerned with uncovering the implicit knowledge that modelers rely on when designing models. This knowledge is encoded in clearly defined rules for transforming models. The existence of such rules would facilitate later changes to the system, e.g. to fix errors or to extend the functionality [37]. This knowledge, which is now made explicit, can be further reused in other projects where similar models are being used. Furthermore, reuse can be enhanced by further abstracting the transformation rules and expressing them in terms of meta-models, making them more general and better applicable to a wider range of situations.

Models need to express certain characteristics to be suitable for MDD. Completeness [18], executability [24], inexpensiveness [6], and understandability [39, 24, 52] are only a few of the recurring ones. However, MDD includes more than just the models themselves. Meta-models, transformations between models, a process for creating and managing the models and their transformations, as well as tools to support the process, are all necessary for an MDD approach to serve its purpose [29]. Transformations enable a controlled transition between the phases of the development process, moving from one abstraction to another, and adding or hiding details to make the design executable. When defined between models developed in the same language, the transformations are based on the language of the models. However, when models are developed in different languages, the transformations need to refer to the meta-models.

MDD is still a new development approach with many unresolved issues which influence its adoption. Developing a repository of MDD models is a complex and difficult task, contributing to only a few repositories being developed. This in turn hinders MDD projects' ability to create, manage, and reuse models [19]. In addition, MDD generated code can be less efficient than manually-written code [52]. The lack of mechanisms for incorporating model changes without regenerating the whole system affects the scalability and suitability of MDD for large projects [52]. Furthermore, transformations based on meta-models can lead to vertically isolated modeling environments, limiting the general applicability of models and transformations, and contributing to the very problem which MDD attempts to alleviate [37].

In this paper, Model-Driven Engineering (MDE) is also referred to as MDD, since they both refer to the same set of core development principles, as found in [6, 19, 21, 29, 37, 53].

3 Current Approaches to Requirements Integration in MDD

Attempts to explicitly use requirements in model design follow different paths. However, they can be classified in three major groups: some approaches rely on Natural

Language Processing (NLP) to produce a requirements model, which then forms the initial MDD model. Other approaches define guidelines for directly creating such a requirements model, rather than using NLP. Finally, a small group of approaches utilize traceability links between requirements and models (or model elements). This section explores the common characteristics, benefits, and problems associated with each group.

3.1 NLP-Based Integration Approaches

Using natural language for capturing requirements enables different stakeholders to understand, validate, and even actively participate in creating the requirements specification. However, the inherent ambiguity of natural language can cause models to differ from the real intentions of the stakeholders. Some approaches solve this problem using automatic text-processing techniques, producing requirements models which then form the input to MDD. Requirements models generated through NLP replace the analyst-driven model design in Fig. 1.

Text can undergo *syntactical analysis* to find candidate concepts and relationships, without the need to delve into the meaning. Such analysis is called surface (shallow) analysis in [23], and is found in the Linguistic assistant for Domain Analysis (LIDA) methodology [44]. In LIDA, a tool enables users to graphically populate a domain model from lists of candidate classes and attributes, which are generated using a part-of-speech tagger. An additional module in the tool generates a hyper-linked textual description of the created model, which can be used for validating the correctness of the model and explaining it to non-modelers.

Semantic analysis of text, or deep analysis [23], can potentially produce complete models. Several approaches combine syntactical and semantic text analysis to obtain models from natural language requirements. The method presented in [38] generates conceptual models through syntactical analysis of the text to label words with their part of speech, followed by semantic analysis to eliminate possible ambiguities. The result is formally-encoded class diagram of the future system. Moreover, the syntactical, semantic, and pragmatic information of requirements are analyzed in [9] using UniFrame, which is a methodology for assembling distributed applications from heterogeneous components [10]. The discovered information is stored in a knowledge base, which is automatically transformed and integrated with other formal platform-specific knowledge. The supporting Specification Development Environment (SDE) allows users to improve the requirements specification and enter missing information where needed.

Manually preprocessing textual requirements, to prepare the natural language for automatic processing, is common in [9, 38, 44]. However, the combined approach in [23] covers any natural language text by using part-of-speech tagging, enriched with statistical analysis and discourse interpretation. The approach is supported by a tool called Class Model Builder (CM-Builder), but it suffers from being focused on static aspects only. The four-stage approach in [25] also offers full natural language support. Text analysis is based on the relation-triad, where a sentence is composed of a subject, a predicate, and an object. Other parts of speech are also handled, and complex sentences are supported since predicates are verb groups and subjects and objects are noun groups.

The rapid prototype development proposed in [20] combines MDD with Artificial Intelligence (AI) planning and Component Based Development (CBD). MDD is used in the approach to develop the infrastructure code for the prototype, which can be reused even though the prototype itself will be thrown away. To generate the infrastructure code, NLP is used to get a model from natural language requirements.

Table 1 summarizes NLP-based approaches to integrate requirements with models, showing the extent of natural language coverage and the used text analysis type.

Table 1. Summary of the NLP-based approaches to integrate requirements and models

	Text Analysis Form		
	Syntactical Analysis	*Semantic Analysis*	*Combined Analysis*
Complete Natural Language		Relation triad: [25]	CM-Builder: [23]
Subset of Natural Language	LIDA: [44]		UniFrame: [9], Conceptual Model: [38]
	(Note: [20] requires a subset of natural languages, but leaves the choice of the text analysis form to the modeler).		

3.2 Guideline-Based Integration Approaches

Many integration approaches consider a requirements model as the initial MDD model, replacing analyst-driven model creation (see Fig. 1). Unlike NLP-based approaches, guidelines are provided for creating the requirements models. The extent to which the MDD process is covered varies from one approach to another. *Edge integration approaches* discuss only creating a requirements model suitable for MDD, without details on the MDD process. Adding to that, *partial integration approaches* provide guidelines for acquiring a requirements model and transforming it into the initial MDD model. Finally, *total integration approaches* propose a complete MDD approach, including native integration of requirements. Table 2 summarizes the guideline-based integration approaches in terms of the used requirements modeling technique.

Edge Integration Approaches. These approaches only provide guidelines for creating a model from textual requirements, and argue that such a model can be used as input to a suitable MDD process.

Some approaches employ UML-like modeling languages which facilitates integration with UML-based MDD approaches. SysML [43] requirements and use case diagrams are combined in [48] to provide the requirements model, while Executable Use Cases (EUCs) [27] are used in [28]. It is argued that EUCs can cover the whole MDD process, starting with user-level requirements and ending with the implementation of software systems. A combination of tables and the Extended Graph Data Model (EGDM) [49] is used in [50] to structure a knowledge base of the requirements, which is then synthesized using the Generic Modeling Environment (GME) [31] into a requirements model. In [7], the use of Design Spaces [30] is extended to enable them to capture requirements for CBD.

Partial Integration Approaches. Minimizing the time dedicated to obtaining the final software product is the goal of all requirements-enhanced MDD approached. In addition to guidelines for building the initial MDD model, partial integration approaches illustrate how MDD can proceed in using the model. Existing MDD approaches form the basis for partial integration.

The OO-Method [46] is representative of MDD in some integration approaches, where a requirements model is the basis for designing the conceptual model of the OO-Method. Manual guidelines for deriving the conceptual model are suggested in [26], where the TRADE requirements engineering framework [56] provides necessary requirements model. The approach in [2] also includes manual guidelines for transforming the Strategic Rationale (SR) model of the i* framework [58] into the conceptual model. The SR model captures actors and their requirements, exposing the rationale for adopting the requirements, and forming a suitable input to MDD.

To avoid manual tasks, the method in [34] grounds the SR model in a formal specification inspired by KAOS [13]. The resulting formal SR model can be automatically transformed into the OO-Method conceptual model. Nevertheless, the SR model itself is manually created. SR models in [33] are automatically transformed to architectural models in the ACME Architectural Description Language (ADL), using horizontal and vertical transformation rules. Interaction requirements, which specify how users interact with the system, are specifically addressed in [45]. The requirements are captured using sketches, which are common in the human-computer interaction (HCI) community. Formalizing the sketches in ConcurTaskTree (CTT) enables the OO-Method presentation model to be automatically derived from them.

Some approaches for extending MDD with requirement models set the effort to broaden the scope of MDD through different adoptions of Aspect Oriented System Development (AOSD) [8]. Combining AOSD with MDD can provide the necessary mechanisms for separating crosscutting concerns in models [1]. In [54], predefined MDD transformations produce an Aspect Oriented (AO) architectural model from a UML model of the textual requirements. The UML model adheres to a custom-built UML Profile, and captures the requirements as a set of scenarios. Concern-Oriented Model Driven Development (COMDD), which is an AOSD-based process [17], is adapted in [15] and [16] to the special natures of functional and non-functional requirements, respectively. Use cases, the NFR framework [11], and scenario-based specification are used to model the two types of requirements. Using xtUML [36] to capture the models enables automatic transition to the following steps of COMDD.

The NFR framework is also used in [12] to model non-functional requirements. The resulting model is designed to complement functional requirements models (such as use cases and class diagrams). Consistency between functional and non-functional requirements is guaranteed by a single vocabulary built using the Language Extended Lexicon (LEL) [32]. In [14], requirements are captured using Scenario Models [47], and a LEL provides the vocabulary for populating the scenarios. A set of transformation rules, implemented in a tool, are used to process the LEL and scenario models and derive an initial class diagram. Both LEL and Scenario Models are based on natural language, enabling stakeholders to understand the modeled requirements and participate in evaluating and improving them.

Table 2. Summary of how guideline-based approaches integrate requirements and models

	Edge Integration	*Partial Integration*	*Total Integration*
i-based models*		[2, 33, 34, 35]	
Text-based models		Sketches: [45]. Use Cases and Scenarios: [12, 14, 15, 16, 26]. (Notes: LEL is used in [12, 14]. NFR Framework is used in [12, 15, 16]).	PIT-RSL: [53]
UML-based models	SysML: [48]. EUCs: [28]	UML: [22, 54]	
Graph-based models	EGDM: [50]		ATRIUM goal model: [40]. Task taxonomy: [55]
Other models	Extended Design Spaces: [7]		

Some MDD approaches are developed specifically to be used within certain areas, and account for the unique characters of those areas. For example, the SR model of i* is used in [35] to capture requirements for a multi-dimensional (MD) conceptual model for a Data Warehouse (DW). UML profiles are designed to capture the SR model, the specific needs of DW modeling, and the resulting MD model. Query/View/Transformation (QVT) transformations are applied to the customized SR model to obtain the MD conceptual model. In Software Product Lines (SPLs), the tight coupling between requirements and product line models is identified as the major problem of existing solutions [22]. Domain engineering produces in SPL the domain-specific architecture and software components that fit into that architecture. Application engineering is concerned with deriving individual applications that fulfill the requirements of a special situation. All possible variation points in a SPL need to be known beforehand, which limits the ability to derive products for requirements that are not directly derivable from the available components and architecture. UML diagrams, Object Constraint Language (OCL) expressions [42], textual use cases, and a custom-built transformation language are employed in [22] to solve this problem and help developers derive products in a flexible and coherent manner.

Total Integration Approaches. Some MDD approaches are specifically developed to capture requirements. Such processes stretch from the requirements stage to the execution stage, creating a comprehensive approach with native requirements support.

For example, ATRIUM (Architecture Traced from requirements by applying a Unified Methodology) [40] is an MDD methodology for supporting the architectural design decisions and rationale during AOSD. Functional and non-functional requirements are captured using ATRIUM goal model, which is a combination of KAOS and the NFR Framework. Architectural design choices are made based on the goal model, and architectural elements are synthesized to create a draft of the final system

description. Following stages of ATRIUM will refine this proto-architecture. The methodology is supported by a tool which facilitates the creation and transformation of the models [41]. Traceability is also supported by exposing traceability links between the models, based on the transformation rules.

ProjectIT is a role/task oriented MDD approach to interactive system development [53]. Requirements are captured in a formal language called ProjectIT-Requirements Specification Language (PIT-RSL). Integration with the other phases of development is attained through a supporting tool, which offers modules for modeling the requirements, designing interaction views, and generating and editing code.

Navigation requirements of Web applications, stemming from the sequential nature of the Web experience, are specifically supported in [55]. The requirements are modeled using a hierarchical taxonomy of the interaction tasks, enabling the definition of temporal relationships. Activity diagrams are used to describe the steps of each task. The taxonomy is transformed, using three semantically distinct mappings, into a navigational model, which indicates how information and functionality is organized in navigational contexts and links. By viewing the requirements model and the navigation model as graphs, graph transformation rules are used to implement the mappings, which are executed using a transformation tool. Traceability is enabled by including the input of the transformation rule and its type in the output. Using a prototype traceability tool, the output graph can be processed to retrieve and display the traceability information in the HTML format.

3.3 Traceability-Based Integration Approaches

A trace can be defined as any evidence, explicit or not, between any two artifacts on the same level of abstraction or on different levels of abstraction or phases of the development process [57]. Traceability between requirements and models enables a retrospective understanding of the rationale behind design decisions and facilitates change propagation. Some integration approaches are based on establishing traceability links after having created the requirements and models. The links support the analyst-driven creation of models (see Fig. 1), rather than replacing it.

A taxonomy of traceability types in SPL is defined in [4], and evolved into an MDD traceability framework in [5]. The aim is to capture the semantics of trace links, instead of a simple indication that artifacts are related. The taxonomy is generic, and can be used to trace requirements to models (or model elements). Another MDD traceability framework [3] provides a basis for tracing the requirements and assessing their fulfillment in the intermediate models and the final product.

4 Properties of Requirements-to-Model Integration

The approaches discussed in the Section 3 rely on different techniques, technologies, and methods for integrating requirements and models. Although they share a common premise of integration, they do not agree on a core set of properties for realizing that. Based on the surveyed approaches, the following integration-relevant properties were elicited and formulated as the requirements that need to be fulfilled to realize requirements integration with models in MDD.

1. *Support requirements which define both static and dynamic aspects of the information system.* Static aspects of a system cover the relevant concepts and their relationships, and the various states through which the system goes. Dynamic aspects are the processes in the system: how they alter the static aspects of the system, and how they transform the system from one state to another. Usually, static and dynamic aspects are interleaved within the requirements, and an effort to separate them is necessary, since they are captured by different models. The presented approaches were divided between those that only support static aspects ([2, 7, 14, 20, 22, 23, 33, 34, 35, 38, 44, 45, 48, 51, 52, 53]), and those that support both static and dynamic aspects ([9, 12, 15, 16, 25, 26, 28, 40, 47]). Static and dynamic aspects are irrelevant for traceability-based approaches.
2. *Support intentional aspects of the Requirements.* Deriving models from requirements involves understanding the goals which the requirements are meant to serve. An integrated MDD approach must include mechanisms that cover the goals behind the requirements, the environment from which they originate, and the environment which will surround the final system. Only approaches based on i* SR model ([2, 34, 33, 35, 40]) provide intentional support, which may pertain to the fact that i* is used in Goal Oriented Requirements Engineering (GORE).
3. *Include explicit model design guidelines for the initial MDD model.* Essentially, any MDD approach should define the models used for capturing different aspects of the intended system. For those MDD approaches that include more than one step of modeling, transformations and dependencies between the models must also be identified. When requirements are concerned, method guidelines for deriving the initial MDD model from the requirements must be supplied, supported by built in functionality of the MMD tool. Most approaches presented in the previous section include such guidelines. Exceptions are: the NLP-based approach in [20], the edge integration approaches in [7, 28, 47], and all traceability-based approaches.
4. *Improve the completeness of models.* Software development can be seen as translating requirements to an application that fulfills the requirements. In MDD, models, forms, and accompanying "code" fragments such as SQL queries, are the development artifacts, so obtaining complete models is essential for obtaining software the delivers the desired results. Any integration approach must include techniques to verify the models and measure the degree of model completeness – how accurately the models represent the requirements. This property complements enabling change propagation; when deviations are discovered in the models, change management techniques are deployed to eliminate them and re-align the models. Only [14, 23, 47] support ensuring model completeness, and that is limited to manual verification by modelers or stakeholders.
5. *Support requirements which define architectural aspects of the information system.* Requirements may convey information that helps in designing the architecture of the desired system. User interfaces, components, and other architectural aspects can be derived from the requirements. While it is not necessarily included in the requirements, it should be addressed when found. Approaches in [7, 9, 22, 33, 40, 51, 52, 53] include mechanisms to account for architectural aspects.
6. *Enable change propagation between requirements and MDD models.* Links between requirements and models enable analyzing the effect of changes in on onto

Table 3. Integration properties in approaches to integrating requirements and models in MDD

Approaches to integrating requirements and models		Static and Dynamic Aspects	Intentional aspects	Model Design guidelines	Model completeness	Architectural aspects	Change propagation	Model reusability	Tool support
NLP-based integration approaches	[9]	x		x		x		x	x
	[20]	x						x	
	[23]	x		x	x				x
	[25]	x		x					
	[38]	x		x					
	[44]	x		x					x
Edge integration approaches	[7]	x				x		x	
	[28]	x							
	[47]	x			x			x	
	[48]	x		x					x
Partial integration approaches	[2]	x	x	x					
	[12]	x							
	[14]	x		x	x				x
	[15, 16]	x		x					
	[22]	x		x		x		x	
	[26]	x		x					
	[33]	x	x	x		x			
	[34]	x	x	x					
	[35]	x	x	x					
	[45]	x		x					
	[52]	x		x		x			
Total integration approaches	[40]	x	x	x		x	x		x
	[51]	x		x		x			x
	[53]	x		x		x	x		x
Traceability-based integration approaches	[3]						x		
	[4, 5]						x		

the other. Forward and backward traceability must be supported. Traceability can be a bi-product of the guidelines for creating the initial MDD model from requirements. Most approaches fail to capitalize on this opportunity despite the existence of model design guidelines. Besides traceability-based approaches, only two total integration approaches ([40] and [53]) offer traceability support.

7. *Enable reusability of MDD models.* Increasing the reusability of software components is a goal of many software development approaches, tools as well as research projects. Reusability is at the heart of Object Oriented (OO) software development, Service Oriented Architecture (SOA), and Software Production Lines (SPL), only to name a few. A requirements-aware MDD approach must enable reusability of models that stem from similar requirements, since reusing models will increase the abstraction and granularity levels of dealing with reuse artifacts. In the case of MDD, creation, storing, retrieving and application of reuse artifacts is paramount because the whole development activity takes place at this level. However, achieving model reusability seems to be an open issue, since only a few of the presented approaches support it [7, 9, 20, 22, 47].
8. *Provide Supporting Tools.* MDD involves extensive development and use of models. Some MDD approaches also require model transformations. This makes tools essential for the practical application of MDD. Integrating requirements with MDD models must also be supported by tools. Less than half of the proposed approaches provide tool that support the suggested integration guidelines [9, 14, 23, 40, 44, 48, 51, 53].

The surveyed approaches are summarized in Table 3, showing which properties are fulfilled by which approach. The summary shows that at most 5 properties are fulfilled at the same time.

5 Conclusion

Extending MDD to start the development process with requirements can enable a smooth, and ideally, automatic transition to design models. The expected benefits are (a) improved and more explicit modeling decisions, (b) facilitated change propagation, as well as (c) increased model reuse. This in turn increases the efficiency of the development process and improves the quality of the models. Hence in MDD, using models and transformations should start in the early stages of development.

We have surveyed a set of current research approaches for integrating requirements and MDD. While many integration attempts have been proposed, a closer look at them reveals the lack a common basis onto which integration is realized. The approaches were distributed over three groups, signifying different integration mechanisms. Except for traceability-based approaches, requirements models were used to associate requirements to other MDD models. Traceability presupposes that requirements and models already exist, and it can be a bi-product of the explicit relationship between requirements and models. Moreover, NLP techniques can support model generation from textual requirements. However, human intervention is still needed to prepare the original text for NLP and to finalize the generated model.

While using requirements models is prevalent, more elaborated meta-models for requirements are needed to design explicit transformation guidelines. Moreover, the

burden of manual work associated with creating the requirements models, whether by following design guidelines or through NLP, can be minimized using proper tools. Currently tool support for integrating requirements and models is still limited.

We have also proposed a set of properties for requirements-models integration. Fulfilling the properties helps an MDD approach to handle requirements and bridge the gap that currently exists between requirements and models. The suggested properties are by no means exhaustive, and further research is necessary to establish their relevance. However, they can provide the kernel around which a more complete integration of requirements and models can be realized.

This survey did not uncover approaches and tools with a proven track record of being used in the industry. Hence more investigation in this area is needed. Our future work will also target extending MDD with suitable techniques for fulfilling all the suggested integration properties and providing a better connection between requirements and models. This will eventually contribute to a more efficient MDD process.

References

1. Aksit, M.: Systematic Analysis of Crosscutting Concerns in the Model-Driven Architecture Design Approach. In: Symposium on How Adaptable is MDA?, Twente, The Netherlands (2005)
2. Alencar, F., Marín, B., Giachetti, G., Pastor, O., Castro, J., Pimentel, J.H.: From i* requirements models to conceptual models of a model driven development process. In: Persson, A., Stirna, J. (eds.) PoEM 2009. Lecture Notes in Business Information Processing, vol. 39, pp. 99–114. Springer, Heidelberg (2009)
3. de Almeida Ferreira, D., Rodrigues da Silva, A.: A Controlled Natural Language Approach for Integrating Requirements and Model-Driven Engineering. In: Fourth International Conference on Software Engineering Advances, ICSEA 2009, pp. 518–523 (2009)
4. Anquetil, N., Grammel, B., Galvao Lourenco da Silva, I., Noppen, J.A.R., Shakil Khan, S., Arboleda, H., Rashid, A., Garcia, A.: Traceability for Model Driven, Software Product Line Engineering. In: ECMDA Traceability Workshop Proceedings, Berlin, Germany, pp. 77–86. SINTEF, Norway (2008)
5. Anquetil, N., Kulesza, U., Mitschke, R., Moreira, A., Royer, J.C., Rummler, A., Sousa, A.: A Model-Driven Traceability Framework for Software Product Lines. Software and Systems Modeling 9, 427–451 (2010)
6. Atkinson, C., Kuhne, T.: Model-Driven Development: A Metamodeling Foundation. IEEE Software 20(5), 36–41 (2003)
7. Baum, L., Becker, M., Geyer, L., Molter, G.: Mapping Requirements to Reusable Components Using Design Spaces. In: Proc. of the IEEE Int'l Conference on Requirements Engineering, ICRE 2000, pp. 159–167. IEEE, Los Alamitos (2000)
8. Brito, I., Moreira, A.: Aspect-Oriented Software Development: an Overview. In: 5th International Conference on Enterprise Information Systems (ICEIS), Angers, France, pp. 531–534 (2003)
9. Bryant, B.R., Lee, B.S., Cao, F., Zhao, W., Gray, J.G., Burt, C.C., Raje, R.R., Olson, A.M., Auguston, M.: From Natural Language Requirements to Executable Models of Software Components. In: Proc. of the Monterey Workshop on Software Engineering for Embedded Systems: From Requirements to Implementation, pp. 51–58 (2003)
10. Bryant, B.R.: Object-Oriented Natural Language Requirements Specification. In: 23rd Australasian Computer Science Conference, ACSC 2000, pp. 24–30 (2000)

11. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Springer, Heidelberg (1999)
12. Cysneiros, L., do Prado Leite, J.C.S.: Nonfunctional Requirements: from Elicitation to Conceptual Models. IEEE Transactions on Software Engineering 30(5), 328–350 (2004)
13. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal Directed Requirements Acquisition. In: Science of Computer Programming, vol. 20, pp. 3–50. North-Holland, Amsterdam (1993)
14. Debnath, N., Leonardi, M.C., Mauco, M.V., Montejano, G., Riesco, D.: Improving Model Driven Architecture with Requirements Models. In: Fifth International Conference on Information Technology: New Generations, ITNG 2008. pp. 21–26 (2008)
15. Fatwanto, A., Boughton, C.: Analysis, Specification and Modeling of Functional Requirements for Translative Model-Driven Development. In: International Symposium on Knowledge Acquisition and Modeling, KAM 2008, pp. 859–863 (2008)
16. Fatwanto, A., Boughton, C.: Analysis, Specification and Modeling of Non-Functional Requirements for Translative Model-Driven Development. In: International Conference on Computational Intelligence and Security, CIS 2008, vol. 2, pp. 405–410 (2008)
17. Fatwanto, A., Boughton, C.: Concern-Oriented Model-Driven Development Framework. In: 19th Australian Conference on Software Engineering, ASWEC 2008, pp. 527–535 (2008)
18. Favre, J.M.: Towards a Basic Theory to Model Model Driven Engineering. In: 3rd Workshop on Software Model Engineering, WiSME 2004 (2004)
19. France, R., Rumpe, B.: Model-Driven Development Of Complex Software: A Research Roadmap. In: Future of Software Engineering, FOSE 2007, pp. 37–54. IEEE Computer Society, Washington, DC, USA (2007)
20. Fu, J., Bastani, F.B., Yen, I.-L.: Model-driven prototyping based requirements elicitation. In: Martell, C. (ed.) Monterey Workshop 2007. LNCS, vol. 5320, pp. 43–61. Springer, Heidelberg (2008)
21. Gašević, D., Djurić, D., Devedićž, V.: Model Driven Engineering. In: Model Driven Engineering and Ontology Development, pp. 125–155. Springer, Heidelberg (2009)
22. Guelfi, N., Perrouin, G.: A flexible requirements analysis approach for software product lines. In: Sawyer, P., Heymans, P. (eds.) REFSQ 2007. LNCS, vol. 4542, pp. 78–92. Springer, Heidelberg (2007)
23. Harmain, H.M., Gaizauskas, R.: CM-Builder: A Natural Language-Based Case Tool for Object-Oriented Analysis. Automated Software Engineering 10, 157–181 (2003)
24. Henkel, M., Stirna, J.: Pondering on the key functionality of model driven development tools: The case of mendix. In: Forbrig, P., Günther, H. (eds.) BIR 2010. Lecture Notes in Business Information Processing, vol. 64, pp. 146–160. Springer, Heidelberg (2010)
25. Ilieva, M.G., Ormandjieva, O.: Models Derived from Automatically Analyzed Textual User Requirements. In: Fourth International Conference on Software Engineering Research, Management and Applications, pp. 13–21 (2006)
26. Insfrán, E., Pastor, O., Wieringa, R.: Requirements Engineering-Based Conceptual Modeling. Requirements Engineering 7, 61–72 (2002)
27. Jørgensen, J.B., Bossen, C.: Executable Use Cases: Requirements for a Pervasive Health Care System. IEEE Softw. 21(2), 34–41 (2004)
28. Jørgensen, J., Tjell, S., Fernandes, J.M.: Formal Requirements Modeling with Executable Use Cases and Coloured Petri Nets. Innovations in Systems and Software Engineering 5, 13–25 (2009)
29. Caskurlu, B.: Model driven engineering. In: Butler, M., Petre, L., Sere, K. (eds.) IFM 2002. LNCS, vol. 2335, p. 286. Springer, Heidelberg (2002)

30. Lane, T.G.: Studying Software Architecture through Design Spaces and Rules. Tech. Rep. CMU/SEI-90-TR-18, Carnegie Mellon University (1990)
31. Ledeczi, A., Maroti, M., Bakay, A., Karsai, G., Garrett, J., Thomason, C., Nordstrom, G., Sprinkle, J., Volgyesi, P.: The Generic Modeling Environment. In: Workshop on Intelligent Signal Processing (2001)
32. Lindstrom, D.R.: Five Ways to Destroy a Development Project. IEEE Software 10(5), 55–58 (1993)
33. Lucena, M., Castro, J., Silva, C., Alencar, F., Santos, E., Pimentel, J.: A model transformation approach to derive architectural models from goal-oriented requirements models. In: Meersman, R., Herrero, P., Dillon, T. (eds.) OTM 2009 Workshops. LNCS, vol. 5872, pp. 370–380. Springer, Heidelberg (2009)
34. Martínez, A., Castro, J., Pastor, O., Estrada, H.: Closing the Gap between Organizational Modeling and Information System Modeling. In: The Sixth Workshop on Requirements Engineering (WER 2003) (2003)
35. Mazón, J.N., Pardillo, J., Trujillo, J.: A Model-Driven Goal-Oriented Requirement Engineering Approach for Data Warehouses. In: ER Workshops, pp. 255–264 (2007)
36. Mellor, S.J., Balcer, M.J.: Executable UML: a Foundation for Model-Driven Architecture. Addison-Wesley, Indianapolis (2002)
37. Mellor, S.J., Clark, A.N., Futagami, T.: Model-Driven Development. IEEE Software 20, 14–18 (2003)
38. Montes, A., Pacheco, H., Estrada, H., Pastor, Ó.: Conceptual model generation from requirements model: A natural language processing approach. In: Kapetanios, E., Sugumaran, V., Spiliopoulou, M. (eds.) NLDB 2008. LNCS, vol. 5039, pp. 325–326. Springer, Heidelberg (2008)
39. Moody, D.L., Shanks, G.G.: Improving the Quality of Data Models: Empirical Validation of a Quality Management Framework. Information Systems (IS) 28(6), 619–650 (2003)
40. Navarro, E.: Architecture Traced from Requirements applying a Unified Methodology, PhD thesis, Computing Systems Department, UCLM (2007)
41. Navarro, E., Cuesta, C.E.: Automating the trace of architectural design decisions and rationales using a MDD approach. In: Morrison, R., Balasubramaniam, D., Falkner, K. (eds.) ECSA 2008. LNCS, vol. 5292, pp. 114–130. Springer, Heidelberg (2008)
42. Object Management Group (OMG) Object Constraint Language (OCL) 2.0, http://www.omg.org/spec/OCL/2.0/
43. Object Management Group (OMG) Systems Modeling Language (SysML), http://www.omg.org/spec/SysML/
44. Overmyer, S., Benoit, L., Owen, R.: Conceptual Modeling Through Linguistic Analysis Using LIDA. In: Proceedings of the 23rd International Conference on Software Engineering, ICSE 2001, pp. 401–410 (May 2001)
45. Pastor, O., España, S., Panach, J.I., Aquino, N.: Model-Driven Development: Piecing Together the MDA Jigsaw. Informatik-Spektrum. Special issue on Modeling 31(5), 394–407 (2008)
46. Pastor, O., Gómez, J., Insfrán, E., Pelechano, V.: The OO-Method Approach for Information Systems Modeling: from Object-Oriented Conceptual Modeling to Automated Programming. Information Systems 26(7), 507–534 (2001)
47. do Prado Leite, J.C.S., Hadad, G.D.S., Doorn, J.H., Kaplan, G.N.: A Scenario Construction Process. Requirements Engineering 5, 38–61 (2000)
48. dos Santos Soares, M., Vrancken, J.L.M.: Model-Driven User Requirements Specification Using SysML. JSW 3(6), 57–68 (2008)

49. Sanyal, A., Choudhury, S.: An Object-Oriented Conceptual Level Design for Web Data Model. In: Proceeding of International Conference on Methods and Models in Computer Science, ICM2CS 2009, pp. 1–6 (2009)
50. Sanyal, A., Basu, S.S., Choudhury, S.: A Requirement Framework for Enablement of Automatic Generation of Domain Model. In: 2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM), pp. 359–364 (2010)
51. Schmidt, D.C.: Model-Driven Engineering. Computer 39, 25–31 (2006)
52. Selic, B.: The Pragmatics of Model-Driven Development. IEEE Software 20, 19–25 (2003)
53. da Silva, A.R., Saraiva, J., Ferreira, D., Silva, R., Videira, C.: Integration of RE and MDD Paradigms: The ProjectIT Approach and Tools. IET Software 1(6), 294–314 (2007)
54. Sánchez, P., Magno, J., Fuentes, L., Moreira, A., Araújo, J.: Towards MDD transformations from AO requirements into AO architecture. In: Gruhn, V., Oquendo, F. (eds.) EWSA 2006. LNCS, vol. 4344, pp. 159–174. Springer, Heidelberg (2006)
55. Valderas, P., Pelechano, V.: Introducing Requirements Traceability Support in Model-Driven Development of Web Applications. Information and Software Technology 51(4), 749–768 (2009)
56. Wieringa, R.: TRADE: Techniques for Requirements and Design Engineering of Software Systems. Tech. rep., Computer Science Department, University of Twente, Enschede, The Netherlands (1999)
57. Winkler, S., Pilgrim, J.: A Survey of Traceability in Requirements Engineering and Model-Driven Development. Softw. Syst. Model. 9, 529–565 (2010)
58. Yu, E.S.K.: Modelling Strategic Relationships for Process Reengineering, PhD Thesis, University of Toronto, Toronto, Canada (1995)

A Case Study on a GQM-Based Quality Model for a Domain-Specific Reference Model Catalogue to Support Requirements Analysis within Information Systems Development in the German Energy Market

José M. González[1], Peter Fettke[2], H.-Jürgen Appelrath[1], and Peter Loos[2]

[1] OFFIS Institute for Information Technology, R&D Division Energy
Escherweg 2, 26121 Oldenburg, Germany
{jose.gonzalez,appelrath}@offis.de
http://www.offis.de

[2] Institute for Information Systems at the German Research Center for Artificial Intelligence, DFKI GmbH
Stuhlsatzenhausweg 3, Building D3 2, 66123 Saarbrücken, Germany
{peter.fettke,peter.loos}@dfki.de
http://iwi.dfki.de

Abstract. Within this contribution, an approach on a goal-question-metric (GQM) based quality model for a domain-specific reference model catalogue is introduced. First of all, we motivate and present an ontology-based reference model catalogue to support requirements analysis within information systems development in the German energy market. For this purpose, we describe requirements for creating such a catalogue. Based on these requirements, quality metrics, to continuously measure the quality of the catalogue during its development and extension, are presented. In addition, the application of these metrics is shown.

1 Introduction

In the field of information systems development, conceptual (information) modelling has been known for many years. It is often applied to analyse, design and implement information systems. As the process of modelling is in general time-consuming and faulty, the concept of reference modelling was introduced. Here, it provides blueprints to improve and accelerate the modelling process. In addition, it aims at reducing modelling risks and costs to prevent failure of modelling projects. In this regard, model quality is regarded as one major issue within reference model development. The term reference model is not clearly defined within literature, see e.g. analysis in [27], [4], [30], [31] and [10]. Therefore several reference models exist for different stakeholders and purposes, like the Y-CIM [27] or the Zachman framework [36]. Within this contribution, a reference model is regarded as a blueprint that can be used to create a specific model

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 357–371, 2011.

in the context of information systems development or evolution. Therefore, a reference model is not regarded as an attribute of a model but as a relationship between two models [5].

At the time of the development of more and more reference models, identification and selection of adequate or relevant models becomes difficult. Therefore, [12] introduced the concept of a reference model catalogue providing an overview on reference models. As with reference models, quality of reference model catalogues is crucial, too. Current approaches on constructing reference model catalogues or classifications of reference models like [12] or [3] emphasise the maintenance process but lack describing detailed measures and metrics for improving quality.

The objective of this paper is to show how a goal-question-metric (GQM) based quality model can be used to improve the quality of an ontology-based reference model catalogue. This reference model catalogue aims at facilitating requirements analysis within the development of information systems in the German energy market. Therefore, common quality requirements of different domains dealing with models such as (reference) modelling and software or ontology engineering are considered to build a quality model. Application of the quality model is done and experiences within its use are described. In addition, the adaption of the concept of a reference model catalogue to support requirements analysis within the development of information systems for the German energy market is shown.

The remainder of this contribution is organised as follows. Section 2 provides some background information on the German energy market and challenges of software product managers. Afterwards, the concept of a reference model catalogue (RMC) is introduced. Based on this, Sect. 3 describes the motivation and usage scenario of the energy reference model catalogue (Energy RMC) as well as its components. In Sect. 4, a GQM based quality model is introduced. Section 5 shows an extract of the metrics calculated. Finally, Sect. 6 concludes with a summary and an outlook on future work for the problem addressed.

2 Background

2.1 The German Energy Market

The German energy market is facing several challenges due to changes in regulation, technical advancements as well as increasing energy costs and climate achievements like CO_2 reduction, see e.g. [1], [23] or [29]. On the one hand, laws have been approved to encourage competition in the German energy market like the legal unbundling as described in the German energy market act (Energiewirtschaftsgesetz EnWG). On the other hand, technical advancements lead to new products and services like Automated Meter Reading (AMR) and Demand Side Management (DSM). Due to new distributed generation facilities like wind power plants or fuel cells, energy is fed into the grid at different voltage levels and by different producers - former customers having their own generation

can now both act as consumers and producers which feed into the utilities' grid [18].

Current information systems in the German energy market were built to address requirements of the past de facto monopoly environment. Today, companies in the energy market face more competition and have to provide new products, more interfaces and services at lower costs. This requires current information systems to become more flexible and to be able to adapt faster to the evolving business requirements. Both utility companies as well as software developing companies have to deal with these changes and need to adapt their information systems or software products. In this context, requirements analysis plays an important part, but is usually time consuming. The use of reference models and other sources containing domain knowledge seems reasonable to speed up requirements analysis, see e.g. [30], [27] or [21].

In the energy market, several heterogeneous information sources like IT standards, descriptions of information systems, enterprise specific models or laws and regulations exist. They provide valuable domain knowledge to support business driven information systems development. Based on a desk research supplemented by several discussions with domain experts, over 60 sources were identified. Analysing these sources revealed, that they were developed by different organisations, associations, consultants and software vendors, thus often using specific vocabularies and pursuing different goals. Further on, they either focus on one area of the supply chain (like market operations) for several viewpoints (like data, processes or functions) in detail or one viewpoint for several areas (like generation and distribution) [17]. In the following, the term *information sources* is used to categorise all documents providing valuable information to enable a business driven information systems development. Currently, there is no systematic access to information sources available to support business driven development of information systems in the energy market [17]. Identifying suitable sources for use within requirements analysis is therefore difficult and time-consuming. For structuring and providing an overview on reference models, the concept of a reference model catalogue was developed by [11] and is introduced in Sect. 2.3.

2.2 Challenges for Software Product Managers in the Energy Market

As stated before, several factors are currently influencing the German energy market and leading to process changes, more competition as well as increasing market participants and data exchange. This leads to new or changed requirements regarding functionality and security of information systems [18]. Software product managers in energy and software developing companies in charge of driving the functional development of information systems have to deal with these challenges.

Within this contribution, a software product manager or application system responsible is regarded as a person working at a software vendor or company in charge of the development of the business functionality of information systems,

which support energy specific functions like SCADA[1] *or EMS*[2] *systems.*[3] *General or generic IT systems like systems for human resources or technical integration solutions are out of scope.*

On the one hand, delivering high quality information systems at low cost and time to market is crucial. On the other hand, continuous requirements analysis is necessary but time consuming and therefore competing with time and cost goals.

Because of the multitude of heterogeneous information sources using their own specific vocabulary, identifying relevant sources is time consuming. Apart from this, regulatory changes require software product managers to adapt their information systems on a regular basis. This forces software product managers not only to analyse several sources, but also to identify, which functional components of information systems might be affected. These tasks are usually done by business and information technology experts collaboratively.

2.3 The Concept of a Reference Model Catalogue

Fettke and Loos developed the concept of a reference model catalogue (RMC) aiming at providing a systematic and structured overview on reference models. In literature, the terms overview of reference models and reference model catalogue are used as synonyms. According to Fettke and Loos [11], a reference model catalogue is typically structured as a table composed of three columns. The first column provides a structure for classifying reference models (structure part), the second contains the names and authors (main part) and the third one lists several attributes (like modelling language) of the classified reference models (access part).

Several catalogues exist for different purposes and branches, where some also include other sources than reference models, see [10]. This seems reasonable as the term reference model is not precisely defined and other information sources provide valuable information, too.

3 The Energy Reference Model Catalogue

In the following, only a brief introduction of the energy reference model catalogue (Energy RMC) is given, for further details see [17] and [18].

3.1 Motivation and Requirements

The reference model catalogue concept as briefly introduced in Sect. 2.3 is domain independent and therefore tends to use high-level classification and general

[1] Supervisory Control and Data Acquisition.
[2] Energy management system.
[3] Later, the terms software product manager refers to both software product manager and application system responsible.

attributes like modelling language or purpose. To address specific needs of software product managers within the German energy market, detailed classification and specific attributes are needed. The energy reference model catalogue's objective is to provide a domain-specific catalogue considering the needs of software product managers in the German energy market.

This requires a detailed functional structuring of the structure part of the catalogue and a linkage to logical applications supporting specific functions within the energy market. Such a linkage should support the identification of relevant functional parts for companies when analysing their specific applications and hence, identify functional overlaps and relevant information sources.

In contrast to other sectors, the German energy market defines several market roles like distribution system operator or balancing responsible party and prescribes corresponding business functions. Legislation regulates which market roles can be performed in combination by one company and at which conditions. For example, a distribution system operator in Germany is normally not allowed to act as a power plant operator. Therefore, linking market roles to the functional structure part is important, too.

Current overviews of information sources relevant to information systems development in the energy market focus on regulations [6] and IT Standards [32]. These overviews provide a starting point for analysis from specific perspectives, but lack to integrate information sources from various other important viewpoints. In addition, specific methods for construction and maintenance considering software product managers requirements in the energy market are not provided. Within this, continuous analysis of the RMC catalogue's quality is considered a major issue to ensure usefulness and hence support catalogue users.

3.2 Use Case for the Energy RMC

The energy reference model catalogue (Energy RMC) aims at supporting software product and product portfolio managers in the German energy market, see also [17] and [18]. Software product portfolio managers are in contrast to software product managers in charge of the development of several information systems. Information systems as depicted in Fig. 1 are often influenced by several factors. Some are external, like regulation or technological advancements, and some internal, such as requirements of different business departments.

Software product managers are in charge of the development of these systems, especially regarding business functionality. In this context, the Energy RMC provides a core knowledge base where different knowledge sources are stored and integrated. The Energy RMC is in constant development as information sources are subject of change. Within use, construction and maintenance of the Energy RMC, the following actors have to work together, see Fig. 1:

- **Software product and product portfolio managers:** use the RMC as one core knowledge base to identify relevant information sources, perform impact analysis (e.g. in case of regulatory changes) and integrate different sources. For this purpose product and portfolio managers can perform queries

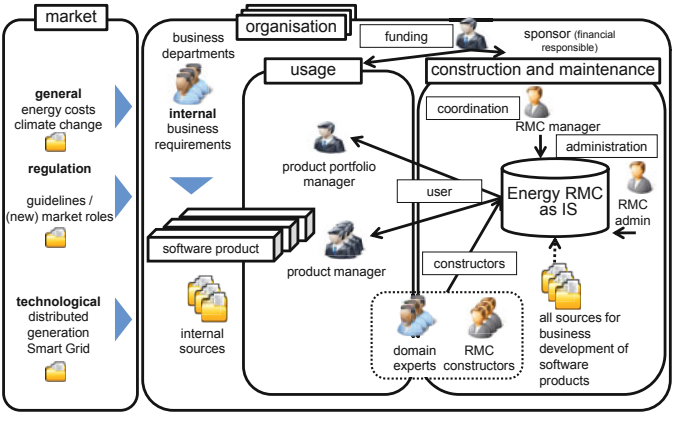


Fig. 1. Usage scenario of the Energy RMC

on the knowledge base. Hereby, they are supported by RMC developers with ontology engineering knowledge.

- **Domain experts:** have practical experience and knowledge regarding particular information sources.
- **RMC-Developer:** support product managers and domain experts when dealing with the ontology-based knowledge base. For example during integration of sources or development of queries.
- **RMC-Model Manager:** coordinates the Energy RMC development and is in charge of the RMC quality.
- **RMC-Admin:** assumes the administration of the RMC.
- **Sponsor:** provides the necessary funding for usage and continuously development of the RMC and supervises its costs.

For the organisational implementation of the Energy RMC, two basic scenarios are possible. First, use, construction and maintenance of the catalogue is done within one company or an affiliated group of companies. Second, construction and maintenance of the catalogue is done by external organisations or service providers and used within different companies.

3.3 Components of the Energy RMC

The Energy RMC aims at supporting the identification of suitable information sources within information systems development in the German energy market, especially regarding requirements analysis.

Therefore, the Energy RMC comprises five components to enable an adequate classification and identification of information sources, see Fig. 2.

First, a functional reference model (FRM) providing a five level hierarchy (supply chain elements to functions) hereby enabling a detailed classification of sources according to functional areas. Second, information sources which optionally can be modelled also as hierarchy (using function groups and functions)

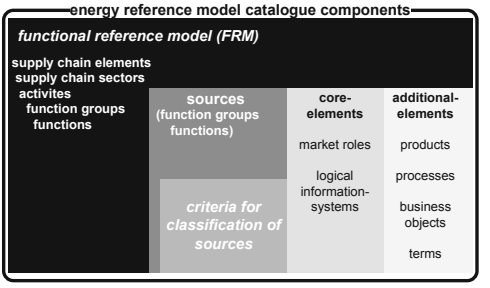


Fig. 2. Components of the Energy RMC based on [17]

and which are classified according to the FRM. Depending on the importance of the source, a linking on different levels of the FRM hierarchy is possible. Third, further classification criteria of sources like coverage - providing information on layers and fields addressed, type - describing the origin and type of sources, or status - indicating the current state of development and the assumed usage [18]. Fourth, core elements, which are of high importance and therefore all logical applications and market roles, should be considered. And finally, additional elements that act as supporting elements to achieve functional completeness. Both core and additional elements (like named market roles or processes) are linked to the FRM. Whereas core elements are linked on the most detailed level (functions), additional elements might be linked on higher abstraction levels like activities. This differentiation together with the linking of sources according to their importance level enables an economic development of the catalogue. Regarding the components of the catalogue concept described in Sect. 2.3, the three main components of the Energy RMC are: FRM, sources and criteria. These components relate to the structure, main and access part of the RMC concept defined by [11]. In addition, core and additional elements extend the RMC concept of [11]. The Energy RMC is formalised as ontology (RMC ontology) using the ontology editor Protégé and the web ontology language (OWL). Furthermore, the sublanguage OWL DL is used because of its maximum expressiveness while retaining computational completeness and decidability.

4 Quality Model

As mentioned in Sects. 2 and 3.1, quality of models and hence of the catalogue are crucial to ensure usefulness and finally lead to high-quality software. In the field of modelling as well as ontology and software engineering, several quality requirements and corresponding metrics exist, for an overview see among others [26], [25], [33], [22] or [7]. The aforementioned topics cover a broad range of different types of models (e.g. data models or process models) and ontologies (e.g. top-level or application ontologies). In addition, different types of models and ontologies address also different purposes (like information systems or organisation development) and users (like IT experts or business analysts). Therefore,

a context specific specialisation is necessary. Quality models or frameworks are often used to enforce quality of models. A quality model or framework groups several quality characteristics, refines them and proposes metrics for assessment to improve the quality of specific models [26]. For instance, ISO/IEC published the ISO/IEC 9261 (software product quality) standard [20] for evaluating quality of software products defining a specific quality model. Taking requirements of various modelling domains as well as ontology engineering into account seems reasonable to develop a quality model for the Energy RMC.

A proven approach for developing quality models and determining metrics is the Goal Question Metric approach (GQM) introduced by Basili et al. [2]. According to [2], the GQM approach consists of three steps focusing on conceptual, operational and qualitative levels. First, goals have to be defined regarding objects of measurement (goals/conceptual level).

In this case, the object of measurement is the Energy RMC and its components. Tables 1 and 2 show a GQM-based quality model for the Energy RMC. Based on literature analysis, 34 sub-requirements were identified and grouped by top-level requirements (R1-R8). Table 1 shows the top-level requirements R1-7 and Table 2 presents R8 (see left side of the tables). The guidelines for modelling (Construction Adequacy, Language Adequacy, Economic Efficiency, Clarity, Comparability and Systematic Design) as defined by [28] where used for further grouping (R1-6, see left side of Table 1). Apart from the guidelines, two additional top requirements were identified. First, Validity (R7, see Table 1 at the bottom), focusing on currency and maturity as well as user acceptance as defined by [34]. Second, Coverage (R8, see Table 2), addressing primarily the analysis of ontological deficits as described by [13]. The sub-requirements SR1-29 focus on analysing only the Energy RMC components, while SR30-34 (Table 2) define requirements to compare the FRM of the Energy RMC with other sources. For each sub-requirement recommended approaches are listed for further analysis, see GOM [28], ISO 9126 [20], Scheer [27], Gomez-Perez et al [16], Fieber et al [14], Van Belle [34], RM (common requirements of reference models as stated for example in [30] or [4]), Moody [26], NASA's software assurance technology center quality model (SATC) [24], Semiotic based ontology evaluation [8], Fenton [9], OntoClean [19] and Fettke & Loos [11]. Furthermore, for each sub-requirement addressed components and viewpoints of the stakeholders are described. Next, questions are provided to characterise the assessment of each goal (questions/operational level). Finally, if possible a metric and its corresponding set of data is provided for quantitative analysis (metrics/quantitative level). In case no quantitative analysis is possible, qualitative measures are described.

Whereas R1-7 (Table 1) primarily focus on analysing the Energy RMC ontology and its components, R8 analysis the coverage of the FRM. Here, the FRM and the hierarchical formalisation of the different information sources integrated into the Energy RMC are compared. This comparison is based on the analysis of ontological deficits described in [13] and [10] and presented in the following.

Table 1. Energy RMC quality model Part 1

		improvement goals			components of the Energy RMC	stakeholder viewpoint	sample questions	sample metrics or measures
		No.	sub-requirement	approaches				
analysis of Energy RMC ontology and its compontents	R1 construction adequacy	SR1	semantic correctness	GOM [28], ISO 9126 [20], Scheer [27], Gomez-Perez et al [16], Fieber et al [14], Van Belle [34]	Energy RMC ontology	Energy RMC-manager	is the Energy RMC described semantically correct?	domain-based semantic analysis
		SR2	syntactic correctness	GOM [28], ISO 9126 [20], Scheer [27], Gomez-Perez et al [16], Fieber et al [14], Van Belle [34]	Energy RMC ontology	Energy RMC-manager	is the Energy RMC described syntactically correct (according the schema)?	**M1:** syntactical non correct elements / all elements
		SR3	completeness	Moody [26], Gomez-Perez et al [16], Fieber et al [14], Van Belle [34]	Energy RMC ontology, FRM, sources, sources criteria, core and additional elements	product manager	does each role/information system is linked to functions? does each function have a link to role/information system? how many elements are linked? how many sources are described in detail/high-level (complete)?	**M2:** linked elements / all elements
		SR4	usability	ISO 9126 [20], Semiotic-based ontology evaluation [8], Gomez-Perez et al [16], Van Belle [34]	Energy RMC ontology	product manager, domain experts, developer, admin, manager	does the Energy RMC fulfil the requirements of the different users?	use case scenario based analysis
		SR5	universal validity	RM [30]/[4], Fieber et al [14], Van Belle [34]	FRM	product manager, manager	are listed functions of the FRM of the energy-Energy RMC linked to at least 2 sources?	**M3:** correct functions / all functions
		SR6	recommendation character	RM [30]/[4], Van Belle [34]	Energy RMC ontology	product manager, sponsor	how many times is the Energy RMC used/cited?	M4: literature/project analysis and count (not really applicable because Energy RMC is not well known and not public available)
	R2 language adequacy	SR7	relevant information	GOM [28]	Energy RMC ontology, schema	product manager, sponsor	are all relevant information for users and constructors modelled?	domain-orientated analysis
		SR8	adequate modelling language	GOM [28], ISO 9126 [20], Fieber et al [14]	Energy RMC ontology, schema	domain experts, developer, admin	is the modelling language adequate?	criteria based analysis
	R3 economic efficiency	SR9	positive cost-benefit ratio	GOM [28]	Energy RMC ontology	sponsor, product manager	what is the cost-benefit ration?	use case scenario based analysis per organisation/project
		SR10	ability to adapt (flexibility, expandability)	ISO 9126 [20], Scheer [27], Fieber et al [14], Moody [26], Van Belle [34]	Energy RMC ontology	sponsor, product manager, domain expert, developer, manager, admin	is the Energy RMC adaptable?	criteria based analysis
		SR11	efficiency	ISO 9126 [20], Van Belle [34]	Energy RMC ontology	sponsor, product manager, domain expert, developer, manager, admin	is the use of the Energy RMC efficient?	use case scenario based analysis per project comparison regarding other approaches.
		SR12	maintainability	SATC [24], Van Belle [34]	Energy RMC ontology	sponsor, manager, admin	is the Energy RMC maintainable?	**M5:** Maintainability index, composed metric possible considering size, unique elements, reusability, modularity
		SR13	volatility	SATC [24]	Energy RMC ontology	product manager, manager	how many elements change per version?	**M6:** average number of elements changed per version
		SR14	reusability	SATC [24]	Energy RMC ontology	manager	how many redundant elements available?	**M7:** number of non unique elements
		SR15	usage costs (introduction, learning, usage)	Van Belle [34]	Energy RMC ontology	sponsor, product manager, developer, manager, admin	how much does the usage of the Energy RMC cost?	use case scenario per organisation/project depends on level of detail and currency of the Energy RMC.
	R4 clarity	SR16	simplicity	GOM [28], Moody [26], Semiotic-based ontology evaluation [8], Fenton [9], Fieber et al [14], Van Belle [34]	Energy RMC ontology	product manager, domain experts, developer, admin, manager	how complex is the Energy RMC? how many elements, classes or relationships contains the Energy RMC? how deep is the hierarchy/inheritance level?	**M8:** number of elements (instances, properties, relationships, unique elements, ...), Inheritance, DIT - depth of inheritance tree, NOC - number of children, coupling between elements (relationships)
		SR17	understandability/ documentation	GOM [28], Moody [26], SATC [24], Van Belle [34]	Energy RMC ontology	product manager, domain experts, developer, admin, manager	for how many elements descriptions or comments exists?	**M9:** number of described or commented elements/ all elements
		SR18	conciseness	Gomez-Perez et al, Fieber et al [14], Van Belle [34]	Energy RMC ontology	manager	how many synonyms are used within descriptions or names of elements?	**M10:** analyse synonym usage with GermanNet, count synonyms
	R5 comparability	SR19	adhere to modelling rules/conventions/ specification	GOM [28], ISO 9126 [20], Gomez-Perez et al [16], Fieber et al [14]	Energy RMC ontology	developer, manager	does the Energy RMC obey rules or conventions?	**M11:** elements following conventions (like each function group should have at least two functions) / all elements
		SR20	consistent information model	GOM [28], Semiotic-based ontology evaluation [8], Gomez-Perez et al [16], Fieber et al [14], Van Belle [34]	Energy RMC ontology	product manager, manager	is the Energy RMC ontological consistent?	**M12:** consistency check of the ontology, number of inconsistent elements / consistency check functionality of protégé can be used.
	R6 systematic design	SR21	adequate structure	GOM [28], SATC, Van Belle [34]	Energy RMC ontology	product manager, domain experts, developer, admin, manager	is the structure of the Energy RMC adequate?	**M13:** number of architectural changes over time criteria-based analysis
		SR22	limitation to relevant information	GOM [28]	Energy RMC ontology	product manager, domain experts, developer, admin, manager	does the Energy RMC only contain relevant information?	criteria based analysis, questionnaire with users
		SR23	modularity	Semiotic-based ontology evaluation [8], Fenton [9], Van Belle [34]	Energy RMC ontology	manager, admin	how many modular like components does the Energy RMC contain? Classes without instances are regarded as module.	**M14:** number of modular components number of modular components / all components
		SR24	rigidity	OntoClean [19]	Energy RMC ontology	manager, admin	how many essential elements does the Energy RMC contain?	**M15:** number of essential elements / all elements
		SR25	identity	OntoClean [19]	Energy RMC ontology	manager, admin	how many identity criteria does the Energy RMC contain?	**M16:** number of identity criteria (attributes) / all attributes
		SR26	unity	OntoClean [19]	Energy RMC ontology	manager, admin	how many bundles of elements that belong together does the Energy RMC contain?	**M17:** number of bundle of elements with relationships / all elements
	R7 validity	SR27	currency	Van Belle [34]	Energy RMC ontology	product manager, manager, domain expert	how up to date is the Energy RMC?	**M18:** average source date, average element change date
		SR28	maturity	Van Belle [34]	Energy RMC ontology	product manager, manager, domain expert	is the Energy RMC mature?	**M19:** analyze number of changes over time
		SR29	user acceptance / popularity	Van Belle [34]	Energy RMC ontology	sponsor, domain expert, product manager	is the Energy RMC accepted or popular?	use questionnaire with stakeholders of the Energy RMC.

Table 2. Energy RMC quality model Part 2

		improvement goals		components of the Energy RMC	stakeholder viewpoint	sample questions	sample metrics or measures
		No. sub-requirement	approaches				
Comparison of FRM to sources	R8 coverage	SR30 incompleteness	Fettke & Loos [11]	FRM and sources	product manager, manager	how incomplete is the FRM? how many functions of the FRM are not referenced by source functions?	**M20:** incompleteness =number of FRM functions without reference to sources (focus models) normalised incompleteness = incompleteness / number of functions of the FRM
		SR31 excess	Fettke & Loos [11]	FRM and sources	product manager, manager	what is the level of excess of the different sources? how many source functions don't have a reference to the FRM?	**M21:** excess=number of source functions without references to FRM functions normalised Excess= Excess / number of source functions
		SR32 overload	Fettke & Loos [11]	FRM and sources	product manager, manager	what is the level of overload of the different sources? how many source functions reference more than one FRM function?	**M22:** overload=number of source functions with references to more than one FRM functions normalised overload= overload / number of source functions
		SR33 redundancy	Fettke & Loos [11]	FRM and sources	product manager, manager	how redundant are the sources? how many source functions reference more than one Energy RMC function?	**M23:** redundancy=number of source functions with more than one reference to the same FRM function normalised redundancy= Redundancy / number of source functions
		SR34 similarity	Van Belle [34]	FRM and sources	product manager, manager	how similar is the FRM to sources?	**M24:** composed metric of SR30-SR33= norm. SR32 + norm. SR33 - norm. SR30 -norm. SR31

The idea of an ontological evaluation was originally proposed by Wand and Weber [35]. Based on this work, Fettke and Loos [13] introduced the concept for evaluation of reference models applying analysis of ontological deficits based on mappings of an ontological construct (named framework model) and a grammar (named focus model), see Fig. 3. With respect to mappings, four ontological deficiencies can be distinguished [13]:

- **Incompleteness (OD1):** Can each ontological construct be mapped onto a construct of the grammar? Otherwise a grammar is incomplete.
- **Redundancy (OD2):** Can each ontological construct be mapped onto exactly one or more than one grammatical construct?
- **Excess (OD3):** Can each grammatical construct be mapped onto an ontological construct?
- **Overload (OD4):** Can each grammatical construct be mapped onto exactly one or on more than one ontological construct?

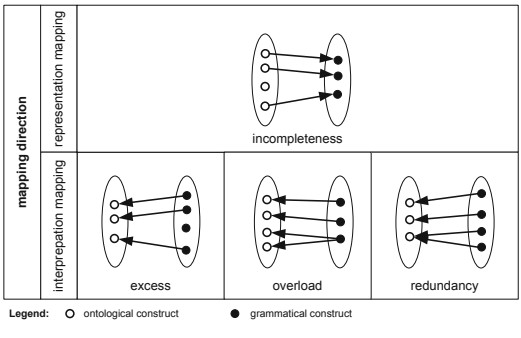


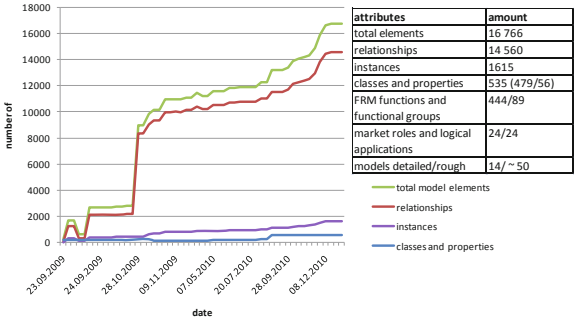
Fig. 3. Ontological deficit analysis by [10]

Adaption of Fettke and Loos approach for the Energy RMC is used to determine the coverage of the FRM with regard to hierarchical modelled sources. Fettke describes in [10] the application of the ontological deficits analysis where the Bunge-Wand-Weber model was used as framework model to analyse other models like the Y-CIM. Here, the FRM is the framework model and the hierarchical structures of the sources act as focus models. To enable comparison of evaluation of different focus models (grammars), normalisation of the listed deficiencies is recommended [10]. For corresponding calculation formulae see metric definitions for SR30-33 in Table 2.

5 Analysing the Energy RMC

In this section, the results of the continuous analysis of the Energy RMC during its development are described and some data on sample metrics are presented.

Figure 4 shows the development of the Energy RMC which started in mid 2008 and is still ongoing. Until now, over 18 versions were provided. The Energy RMC elements continuously increased. Refactoring activities were performed on a regular basis and lead to temporary small decrease of the total amounts of elements. Here, the quality model and its metrics supported quality analysis and the refactoring process. The major part of the total elements is provided by relationships followed by instances. Whereas relationships and instances steadily increased, the amount of classes and properties gained a certain maturity and stayed constant after large changes at the very beginning of the construction process.



attributes	amount
total elements	16 766
relationships	14 560
instances	1615
classes and properties	535 (479/56)
FRM functions and functional groups	444/89
market roles and logical applications	24/24
models detailed/rough	14/ ~ 50

Fig. 4. Development of the Energy RMC and basic data

As mentioned in Sect. 4, the ontological analysis of deficits is considered of major interest for analysing the FRM. Table 3 shows 14 German laws and regulations for electricity and gas that are compared with the current FRM. All sources were formerly hierarchical modelled (in function groups and functions), formalised and linked to the FRM. Based on this mapping, the ontological deficits OD1-OD4 (see Sect. 4) were calculated. The normalised counterpart of OD1 is

calculated by OD1/total amount of functions of the FRM (444) and for OD2-4 by ODx/amount of elements of the corresponding focus model.

The max values of each column are highlighted in black and min values in dark grey. Corresponding electricity and gas regulations are arranged one below the other, see focus models 4/5, 6/7, 8/9 or 10/11 in Table 3. Here similarities in regulation of electricity and gas became obvious (see Table 3) as their metrics results show similar values. The analysis of ontological deficits and its comparison helps in this case to check correct integration of sources or at least provides valuable information or hints for their correct assignment. For example, in a previous version of the Energy RMC, the values of 6 and 7 differed strongly, even though they address the same functional areas for electricity and gas. By comparing the ontological deficits this was revealed and errors/defects corrected.

Table 3. Ontological analysis of the FRM and some regulations (max values in black and min in dark grey)

No.	focus model	total elements	incompleteness	excess	overload	redundancy	norm incompleteness	norm excess	norm overload	norm redundancy
1	EnWG	140	325	110	21	14	0,73	0,79	0,15	0,10
2	EEG	71	399	6	46	59	0,90	0,08	0,65	0,83
3	MessZV	14	418	2	8	10	0,94	0,14	0,57	0,71
4	StromNEV	33	425	3	30	30	0,96	0,09	0,91	0,91
5	GasNEV	35	431	3	28	32	0,97	0,09	0,80	0,91
6	StromGVV	23	434	0	6	20	0,98	0,00	0,26	0,87
7	GasGVV	23	434	0	6	20	0,98	0,00	0,26	0,87
8	StromNZV	33	399	8	18	8	0,90	0,24	0,55	0,24
9	GasNZV	55	343	10	43	29	0,77	0,18	0,78	0,53
10	NDAV	31	429	2	15	28	0,97	0,06	0,48	0,90
11	NAV	31	429	2	15	28	0,97	0,06	0,48	0,90
12	KAV	9	438	2	7	6	0,99	0,22	0,78	0,67
13	KWK-G	16	417	6	9	8	0,94	0,38	0,56	0,50
14	ARegV	45	433	11	28	32	0,98	0,24	0,62	0,71

As shown in Table 3, the regulation providing the highest amount of total elements is number one, the EnWG, the German energy market act, which in fact addresses both electricity and gas and is the most extensive one. At the same time, it provides the maximum values on the attributes excess and norm. excess in Table 3. A deeper analysis revealed that the EnWG was integrated into the Energy RMC at an early stage and in the mean time the FRM evolved so that linking between EnWG and FRM has to be checked and updated. Further on, the metrics M20-24 were used to analyze the coverage of the Energy RMC with regard to specific functional models of information systems of a regional utility company. Here, the proposed metrics proved helpful for localizing coverage deficits within the Energy RMC.

Calculation of the metrics of the quality model for the Energy RMC ontology was done using the Protégé OWL API 3.4. This API already provided basic metric functionality like calculation of classes, instances or properties as well as support for the SPARQL query language. Based on this, custom development to implement the metrics of the quality model was done. Currently the implementation and evaluation of the metrics M20-24 as well as the ones shown in Fig. 4 was done, the implementation of the remaining metrics is still on going.

6 Conclusions and Outlook

In our contribution, we first introduced a reference model catalogue for the German energy market formalised as ontology. Second, we presented a GQM-based quality model for continuous quality improvement and analysis. Finally, the concept of a reference model catalogue [11] as well as the GQM [2] and ontological evaluation approaches [13] were successfully adapted to fit the needs of the Energy RMC.

The presented quality model proved to be helpful for the continuous analysis and the improvement of the catalogue. Here, the calculation of quality metrics specially supported the improvement of the Energy RMC. Using the Protégé API helped during the implementation of the metrics, although to optimise the performance is still necessary.

Furthermore, to support the development and improvement of the catalogue qualitative empirical methods were used. Here, several workshops and interviews with prospective users of the catalogue took place. Also surveys, to gather requirements for supporting software product managers in the German energy market within their requirements analysis, are currently conducted. In addition, further analytical evaluation of the Energy RMC, considering various perspectives as described by Frank [15], is planned. The calculation of metrics was much easier than empirical evaluation which required more effort. Nevertheless, empirical evaluation supported the identification of interesting metrics.

Further research will focus on the use of the Energy RMC in several scenarios, the enhancement of the quality model as well as finalizing the implementation of the metrics. During first projects using the Energy RMC and the quality model as well as discussions with prospective users we found out that users are usually only interested in selected parts of the catalogue. Therefore, prospective users are not interested in high quality for all parts of the catalogue. Instead, they strive for high quality for parts of the catalogue that are of particular interest within their prime work. Therefore, to apply the quality model for assessment of quality regarding selected parts of the catalogue seems economic and reasonable. In addition, the presented ontological metrics can support the ranking and hence identification of information sources as well as analyzing the coverage of the Energy RMC.

The presented quality model aims at analysing quality characteristics of the Energy RMC but considers common quality requirements of various fields, see Sect. 4. Therefore, reuse and adaption of the measures and metrics of the Energy RMC quality model to improve other reference model catalogues seems feasible. Nevertheless, usefulness in other contexts, development of additional metrics and further empirical analysis is still subject to future work.

References

1. Appelrath, H.J., Chamoni, P.: Veränderungen in der Energiewirtschaft - Herausforderungen für die IT (Changes in the energy sector - Challenges for IT). Wirtschaftsinformatik 49(5), 329–330 (2007)

2. Basili, V.R., Caldiera, G., Rombach, H.D.: The goal question metric approach. Encyclopedia of Software Engineering 1, 528–532 (1994)
3. Braun, R., Esswein, W.: Classification of Reference Models. In: Decker, R., Lenz, H.-J. (eds.) Advances in Data Analysis, Proceedings of the 30th Annual Conference of the Gesellschaft für Klassifikation e.V., Freie Universität Berlin, March 8-10. Studies in Classification, Data Analysis, and Knowledge Organization, pp. 401–408. Springer, Heidelberg (2007)
4. vom Brocke, J.: Referenzmodellierung: Gestaltung und Verteilung von Konstruktionsprozessen (Reference modeling: design and distribution of construction processes). Ph.D. thesis, Berlin (2003)
5. vom Brocke, J., Fettke, P.: Referenzmodellierung. In: Kurbel, K., Becker, J., Gronau, N., Sinz, E., Suhl, L. (eds.) Enzyklopädie der Wirtschaftsinformatik. Oldenbourg, München (2009)
6. Bundesverband der Energie- und Wasserwirtschaft (BDEW) (German Federal Association for Energy and Water): Survey of the most important laws, ordinances, specifications, guidelines and recommended action on the subject of electricity business processes (2008)
7. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. IEEE Transactions on Software Engineering (6), 476–493 (June)
8. Dividino, R., Romanelli, M., Sonntag, D.: Semiotic-based Ontology Evaluation Tool S-OntoEval. In: Proceedings of the Sixth International Conference on Language Resources and Evaluation, ELRA (2008)
9. Fenton, N.: Software metrics: successes, failures and new directions. Journal of Systems and Software 47(2-3), 149–157 (1999)
10. Fettke, P.: Referenzmodellevaluation: Konzeption der strukturalistischen Referenzmodellierung und Entfaltung ontologischer Gütekriterien (Evaluation of reference models: Conceptual design of structural reference modeling and development of ontological quality criteria). Ph.D. thesis, Berlin (2006)
11. Fettke, P., Loos, P.: Der Referenzmodellkatalog als Instrument des Wissensmanagements - Methodik und Anwendung (The reference model catalogue a knowledge management tool - Methodology and Application). In: Becker, J., Knackstedt, R. (eds.) Wissensmanagement mit Referenzmodellen. Konzepte für die Anwendungssystem- und Organisationsgestaltung, pp. 3–24. Springer, Berlin (2002)
12. Fettke, P., Loos, P.: Classification of reference models - a methodology and its application. Information Systems and e-Business Management 1(1), 35–53 (2003)
13. Fettke, P., Loos, P.: Ontological Analysis of Reference Models. In: Green, P., Rosemann, M. (eds.) Business Systems Analysis with Ontologies, ch. III, pp. 56–81. Idea Group Publishing, USA (2005)
14. Fieber, F., Huhn, M., Rumpe, B.: Modellqualität als Indikator für Softwarequalität: eine Taxonomie (Model quality as indicator for software quality: a taxonomy). Informatik-Spektrum 31(5), 408–424 (2008)
15. Frank, U.: Evaluation of Reference Models. In: Fettke, P., Loos, P. (eds.) Reference Modeling for Business Systems Analysis, ch. VI, pp. 118–140 (2007)
16. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological engineering: With examples from the areas of knowledge management, e-commerce and the semantic web, 3. print. edn. Springer, London (2004)
17. González, J.M., Appelrath, H.-J.: "Energie-RMK" Ein Referenzmodellkatalog für die Energiewirtschaft (Energy-RMC - A reference model catalogue for the energy sector). In: GI-Modellierung 2010, pp. 319–334 (2010)

18. González, J.M., Uslar, M.: An ontology-based method to construct a reference model catalogue for the energy sector. In: Smolnik, S., Teuteberg, F., Thomas, O. (eds.) Semantic Technologies for Business and Information Systems Engineering: Concepts and Applications (2011)
19. Guarino, N., Welty, C.A.: An Overview of OntoClean. Applied Ontology, 1–20
20. ISO, IEC: ISO/IEC 9126-1 Software engineering - Product quality - Part 1: Quality model (2001)
21. Jarke, M., Loucopoulos, P., Lyytinen, K., Mylopoulos, J., Robinson, W.: The brave new world of design requirements: Four key principles. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 470–482. Springer, Heidelberg (2010)
22. Kaner, C., Member, S., Bond, W.P.: Software Engineering Metrics: What Do They Measure and How Do We Know?. Direct, 1–12 (2004)
23. Kurth, M.: The Changing Structure of the Utility Industry form the Perspective of Regulation Authorities. In: Bausch, A., Schwenker, B. (eds.) Handbook Utility Management, pp. 193–206. Springer, Berlin (2009)
24. Lawrence, E.H., Rosenberg, L.H.: A Software Quality Model and Metrics for Identifying Project Risks and Assessing Software Quality. In: 8th Annual Software Technology Conference (1996), http://satc.gsfc.nasa.gov/support/STC_APR96/qualtiy/stc_qual.html
25. McCall, J., Richards, P., Walters, G.: Factors in software quality. US Rome Air Development Center Report 1-3(NTIS AD/A-049 014) (1977)
26. Moody, D.: Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. Data & Knowledge Engineering 55(3), 243–276 (2005)
27. Scheer, A.-W.: Business process engineering: Reference Models for Industrial Enterprises, 2nd edn. Springer, Berlin (1994)
28. Schuette, R., Rotthowe, T.: The Guidelines of Modeling - an approach to enhance the quality in information models Introduction: Theoretical Aspects in Information Modeling. In: Proceedings of the 17th International Conference on Conceptual Modelling, Singapore, pp. 1–16 (1998)
29. Starace, F.: The Utility Industry in 2020. In: Bausch, A., Schwenker, B. (eds.) Handbook Utility Management, pp. 147–167. Springer, Berlin (2009)
30. Thomas, O.: Management von Referenzmodellen: Entwurf und Realisierung eines Informationssystems zur Entwicklung und Anwendung von Referenzmodellen (Management of reference models: Design and Implementation of an information system to design and use reference models). Ph.D. thesis, Berlin (2006)
31. Thomas, O.: Understanding the Term Reference Model in Information Systems Research: History, Literature Analysis and Explanation. In: Bussler, C., Haller, A. (eds.) Business Process Management Workshops, pp. 484–496 (2006)
32. Uslar, M., Rohjans, S., Bleiker, R., González, J.M., Suding, T., Specht, M., Weidelt, T.: Survey of Smart Grid Standardization Studies and Recommendations - Part 2. In: IEEE SmartGridComm 2010 (2010)
33. Uthayan, K.R.: A Survey on Designing Metrics suite to Asses the Quality of Ontology. Journal of Computer Science 8(8) (2010)
34. Van Belle, J.-P.: A Framework to Evaluate the Quality of Information System Models. Ingénierie des systèmes d'information 9(5-6), 11–29 (2004)
35. Wand, Y., Weber, R.: An Ontological Evaluation of Systems Analysis and Design Methods, pp. 79–107. Elsevier Science Publishers, North-Holland (1989)
36. Zachman, J.A.: A framework for information systems architecture. IBM Systems Journal 38(2.3), 454–470 (1999)

Requirements Evolution: From Assumptions to Reality

Raian Ali[1,2] and Fabiano Dalpiaz[1], Paolo Giorgini[1], and Vítor E. Silva Souza[1]

[1] DISI, University of Trento, Italy
[2] LERO - The Irish Software Engineering Research Centre
{raian.ali,dalpiaz,paolo.giorgini,vitorsouza}@disi.unitn.it

Abstract. Requirements evolution is a main driver for systems evolution. Traditionally, requirements evolution is associated to changes in the users' needs and environments. In this paper, we explore another cause for requirements evolution: *assumptions*. Requirements engineers often make assumptions stating, for example, that satisfying certain sub-requirements and/or correctly executing certain system functionalities would lead to reach a certain requirement. However, assumptions might be, or eventually become, invalid. We outline an approach to monitor, at runtime, the assumptions in a requirements model and to evolve the model to reflect the validity level of such assumptions. We introduce two types of requirements evolution: autonomic (which evolves the priorities of system alternatives based on their success/failure in meeting requirements) and designer-supported (which detects loci in the requirements model containing invalid assumptions and recommends designers to take evolutionary actions).

Keywords: Requirements Engineering, Requirements Evolution, Contextual Requirements, Requirements at Runtime.

1 Introduction

The satisfaction of users' requirements through a developed system is inherently uncertain. Indeed, requirements evolve, and the original system might become inadequate to meet the evolved requirements. Since such evolution is unavoidable, system necessarily has to evolve in order to keep requirements satisfied. Traditionally (e.g., [1,2]), requirements evolution is driven by changes in the users' needs, the operational environment (laws, policies, economical situation), the co-operative systems, and the underlying technology. We explore a new and primitive driver for requirements evolution which is the uncertain validity of the assumptions included in a requirements model.

Requirements are expressed via requirements models. These models contain assumptions, rather than certainties, made by designers about the relation between the system, the requirements, and the environment where the system is to operate. The model may state that a certain requirement will be met by correctly developing and executing specific software functionalities and/or by meeting other sub-requirements. Such assumptions could turn out to be invalid when the system operates. The operation could reveal that some assumptions were initially, or eventually become, invalid. The detection of invalid assumptions necessitates evolutionary actions, which result in a revision of the requirements model to reflect reality. Desirably, these actions are done by the system autonomously. However, the designers' intervention could be often required.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 372–382, 2011.

Our goal in this paper is to enable monitoring the runtime system operation and exploiting it to evolve the requirements models in a lifelong style. We intend to develop systems that support requirements evolution either autonomously or by recommending designers to take evolutionary actions. For example, a shopping mall adminstration intends to build a system to interact with customers' via their PDAs in order to meet the requirement R= "customers head to cash desk when closing time is approaching":

- *Assumptions*. An analyst develops a requirements model stating that R is reached if "customer is notified to leave" (R_1) and "customer is instructed how to leave" (R_2). The model states also that R_2 is reached by one of two software variants: "digital map is shown on customer's PDA" ($SV_{2.1}$) and "customer is tracked and given voice commands via his PDA" ($SV_{2.2}$). These are just assumptions made by the analyst.
- *Autonomic evolution*. As software is deployed, the analyst assumes that $SV_{2.1}$ and $SV_{2.2}$ are equally able to meet R_2. After two months, the operation reveals that $SV_{2.2}$ succeeded in meeting R_2 less often than $SV_{2.1}$. Thus, software should autonomously evolve the requirements model by giving $SV_{2.1}$ higher priority than $SV_{2.2}$ for R_2.
- *Designer-supported evolution*. Software operation could also reveal that reaching R_1 and R_2 does not always lead to reach R. Customers, even if notified and guided on how to leave, still don't leave the mall on time. If such assumption is often invalid, software will ask designers to revise the requirements model and fix it.

We focus on the evolution of contextual requirements models which capture the relation between the state of the environment where the system operates (context [3]) and requirements. Some contexts activate a requirement and others represent preconditions for applying software variants aiming to meet certain requirement. Recently, several contextual requirements models have been proposed to capture such a relationship [4,5,6]. However, modeling contextual requirements is a hard task in which designers need to make assumptions with high uncertainty, such as stating that executing a certain software variant in a specific context will lead to reach a certain requirement.

In this paper, we discuss the evolution of contextual requirements. We articulate the problem of requirements evolution that originates from the invalidity of the assumptions included in contextual requirements (Sec. 2) represented via contextual goal models [5,7] (Sec. 3). We address the two kinds of evolution introduced earlier (autonomic and designer-supported), specifying what information the system has to monitor at runtime and showing how to use this information for evolving the contextual requirements model (Sec. 4). We end the paper with conclusions and future work directions (Sec.5).

2 Requirements Evolution: A Viewpoint

Software systems operate in an environment. The state of such environment, denoted by the notion of *context* [3], is variable. There is a strong mutual influence between context and requirements. This is particularly true in emerging computing paradigms such as ubiquitous and mobile computing, where context-awareness is fundamental for successful software operation. On the one hand, a certain context might *activate* a requirement

or be a *required* precondition for the execution of a software variant designed to reach activated requirements. On the other hand, a requirement corresponds to a *target* context, i.e. the desired state of the environment associated to requirement satisfaction.

In Fig. 1a we depict our general picture about the relation between requirements, software, and context. In Fig. 1b we exemplify it through one specific requirement for a mobile software meant to advertise products to customers in shopping malls. A requirement R is activated—software has to meet it—if its activation context holds. In turn, a context holds if one of its variants holds. A context variant is a conjunction of atomic environmental facts the system can verify. Thus, R is activated if any activation context variant ACV_i is true. R is reached if any of its target context variants TCV_j holds. In order to reach a requirement, a software variant SV_k should be executed. A certain variant is adoptable if any of its required context variants $RCV_{k.j}$ holds.

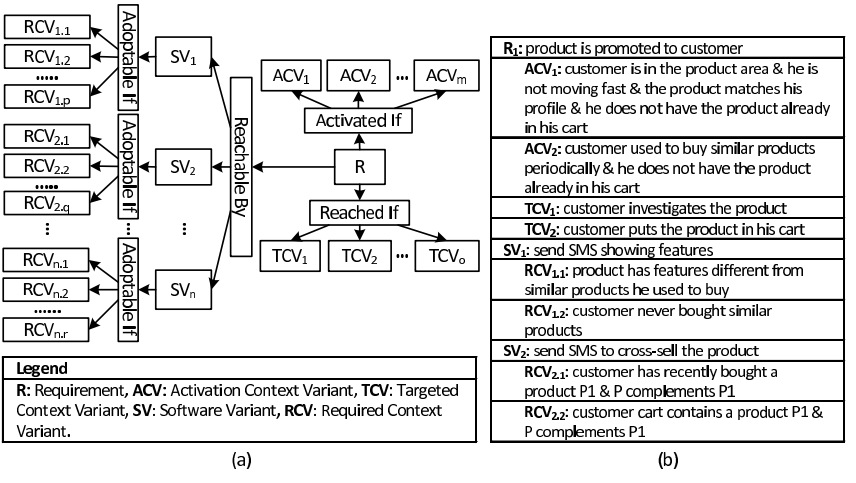


Fig. 1. The relation between Requirements, Context, and Software

Requirements models contain statements that might be, or become, invalid in practice. Thus, a requirements model contains a set of assumptions. These assumptions should be continuously monitored and, if invalid, requirements models should evolve to reflect reality. We list now some types of assumptions that a requirements model could contain. These assumption types are illustrated via examples taken from Fig. 1b:

1. *Activation assumption*. It concerns the hypothesis that a certain context variant activates a requirement. In practice, a requirement might not need to be activated when one of its activation context variants holds. For example, ACV_1 is presumed sufficient to activate R_1, while in practice ACV_1 could miss some additional contextual conditions, and the promotion might lead to a negative customer reaction if those missing conditions are not considered.
2. *Adoptability assumption*. It concerns the hypothesis that a requirement is met by a software variant which is adoptable in a certain set of required context variants.

However, a certain software variant might fail to meet the requirement no matter if one of its required context variants holds. Also, the ability of a software variant to meet a requirement could vary according to each of its required context variants. For example, SV_2 (when $RCV_{2.2}$ holds) could lead to meet R_1 more often than SV_1 (in both of its required context variants $RCV_{1.2}$ and $RCV_{1.1}$).

3. *Refinement assumptions*. It concerns the hypotheses related to the requirements refinements stating that (i) a decomposed requirement is met if all sub-requirements are met; and (ii) a specialized requirement is met if any sub-requirement is met. Suppose R_1 is decomposed into "promote by PDA" ($R_{1.1}$) and "a staff is available for further information" ($R_{1.2}$). Meeting both $R_{1.1}$ and $R_{1.2}$ may not lead to a successful promotion (R_1 is not met). Suppose now a requirements model where R_1 is specialized into "PDA-based promotion" ($R_{1.1'}$) and "staff-based promotion" ($R_{1.2'}$). Meeting $R_{1.1'}$ (e.g. the customer reads the information sent to his PDA) might not lead to a successful promotion (R_1 is not met).
4. *Requirements achievement assumptions*. It concerns the hypothesis that a requirement is satisfied if any of its target context variants holds. In practice, reaching one of these variants may not imply that the requirement is really met. For example, upon executing SV_1 or SV_2, a customer may investigate the product (TCV_1 holds), but this does not necessarily mean he becomes interested in the product.

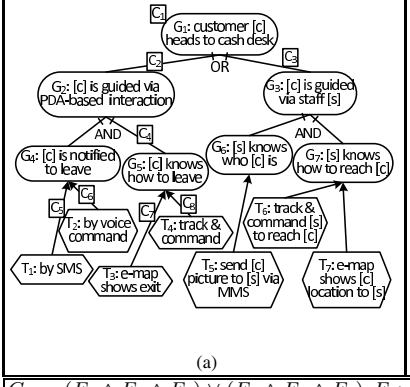
We view *requirements evolution as a continuous movement from assumptions-based requirement to reality-based ones*. The system has to continuously monitor assumptions at runtime and, when an invalid assumption is identified, it is fixed by evolving the requirements model in one of 2 styles; autonomous or designer-supported. Out of the possible combinations between the 4 assumption types and the 2 evolution styles, explained above, we explore 2 combinations; (i) autonomic evolution of adoptability assumptions and (ii) designer-supported evolution of refinement assumptions.

3 Background: Contextual Goal Model

Goal models provide a systematic refinement of user requirements, understood as goals, to derive alternative sets of functionalities software has to support [8,9]. Goals (graphically represented by ovals) can be refined via AND-decomposition or OR-decomposition. In an AND-decomposition, all subgoals should be achieved to reach the decomposed goal. In an OR-decomposition, the achievement of one subgoal is enough to reach the decomposed goal. Goals are ultimately reached by means of executable processes called tasks (represented by hexagons).

Contextual goal models weave together the variabilities of both goal achievement and context [5,7]. Context is specified at a set of goal model variation points. The semantics of context influence differs according to each point. The contexts specified at the variation points Root-goal and AND-decomposition are *activation contexts*; they represent stimulating conditions for goals/tasks. The contexts specified at OR-decomposition, Means-end, and Delegation are *required contexts*; they represent adoptability preconditions for alternatives means to reach/execute goals/tasks.

In this work, we also interpret the satisfaction criteria of a goal as a *target context*, i.e., a state of the world to reach. In Fig. 2a, we depict a contextual goal model example.



$C_1 = (F_1 \wedge F_2 \wedge F_3) \vee (F_1 \wedge F_2 \wedge F_4)$. F_1: closing time is approaching, F_2: customer is far away from all cash desks, F_3: customer is still shopping, F_4: customer is heading far from cash desk
$C_2 = (F_5 \wedge F_6) \vee (F_6 \wedge F_7)$. F_5: customer is using his PDA for different services from time to time recently, F_6: his language is supported currently, F_7: he is familiar with PDA as he stated in the registration form
$C_3 = (F_8 \wedge F_9 \wedge F_{10})$. F_8: staff is close to customer, F_9: staff is free, F_{10}: staff knows a language common to customer
$C_4 = (F_{11} \wedge F_{12})$. F_{11}: customer is visiting the mall for the first times, F_{12}: customer is not familiar with similar mall branches architectures
$C_5 = (F_{13}) \vee (F_{14} \wedge F_{15}) \vee (F_{16} \wedge F_{17})$. F_{13}: customer is often doing some actions on his PDA, F_{14}: customer PDA is not on silent, F_{15}: customer is in a place where ring tone is noticeable with respect to noise level, F_{16}: customer is holding PDA with him, F_{17}: customer PDA vibration is turned on
$C_6 = (F_{18} \wedge F_{19}) \vee (F_{20} \wedge F_{21})$. F_{18}: customer is putting headphones, F_{19}: customer is not listening to loud music/voice on headphones, F_{20}: customer is not making a phone call, F_{21}: customer is not talking to someone
$C_7 = (F_{22} \wedge F_{23}) \vee (F_{22} \wedge F_{24})$. F_{22}: customer has good expertise in navigating e-map, F_{23}: customer's PDA screen is wide enough, F_{24}: the route to a cash desk is simple
$C_8 = (F_{18} \wedge F_{19} \wedge F_{20} \wedge F_{21} \wedge F_{25})$. F_{25}: customer is familiar with direction instructions
$G_1.TC = (F_{26} \wedge F_{27} \wedge F_{28}) \vee (F_{26} \wedge F_{27} \wedge F_{29} \wedge F_{30})$. F_{26}: customer is heading closer to one of the cash desks, F_{27}: customer is not stopping much while heading, F_{28}: customer has not picked a cart, F_{29}: customer has picked a trolley, F_{30}: customer cart is with him.

Fig. 2. An example of a Contextual Goal Model

We have refined the contexts $C_1 \ldots C_8$ and $G_1.TC$ using our context analysis technique (for details see [5,7]), which led to the specifications shown in the table in the bottom of the figure. In Fig. 2b, we show a contextual goal model variant together with its activation and required contexts constructed by accumulating the individual contexts specified at the variation points of each type. Finally, in Fig. 2c, we map the variant shown in Fig. 2b to our view explained in Sec. 2.

4 Evolving Contextual Goal Models

We detail now our viewpoint on contextual requirements evolution by showing how contextual goal models are subject to evolution. First, we describe how to monitor the execution of contextual goal model variants (Sec. 4.1). Second, we explain the basic mechanisms to enact autonomic evolution of adoptability assumptions (Sec. 4.2) and designer-supported evolution of refinement assumptions (Sec. 4.3).

4.1 Monitoring Contextual Goal Models

Monitoring requirements is an essential activity to identify the necessity to evolve the system. Evolution is needed whenever some requirements assumptions prove to be invalid in practice. Monitoring means keeping track of the execution of each software variants and the impact it has on requirements (i.e., are requirements met?). Specifically, we focus on requirements expressed in terms of goals in a contextual goal model. Table 1 exemplifies monitoring of adoptability and refinement assumptions for Fig. 2.

Table 1. Monitoring assumptions: (a) adoptability (b) refinement

(a)

Operation	Enacted Variant	RCV					G_4
		A_1	A_2	A_3	B_1	B_2	
I_1	SV_A	T	F	F	F	T	×
I_2	SV_A	F	T	F	T	T	×
I_3	SV_B	F	T	F	T	T	✓
I_4	SV_A	F	F	T	T	F	✓
I_5	SV_B	F	T	T	F	T	✓
I_6	SV_A	T	T	F	F	T	×

(b)

Operation	G_4	G_5	G_2	G_6	G_7	G_3	G_1
J_1	✓	✓	✓				✓
J_2	×	✓	×				×
J_3	✓	✓	×				×
J_4	✓	✓	✓				×
J_5				✓	✓	✓	×
J_6	✓	×	✓				✓
J_7				✓	✓	✓	✓

In Table 1a, we consider adoptability assumptions taking only the goal G_4 from Fig. 2 (due to space limitations we did not choose the root goal). The goal has two variants; V_A=$\{T_1\}$ and V_B= $\{T_2\}$. Both variants are means to achieve G_4= "customer is notified to leave". There are five required context variants: $RCV_{A_1} = F_{13}$, $RCV_{A_2} = F_{14} \wedge F_{15}$, and $RCV_{A_3} = F_{16} \wedge F_{17}$ for V_A, $RCV_{B_1} = F_{18} \wedge F_{19}$ and $RCV_{B_2} = F_{20} \wedge F_{21}$ for V_B. Every row represents the data collected—via monitoring—during the operation of a specific variant. The columns in the table are an identifier for the operation, an identifier for the enacted variant, the validity of the required context variants, and the satisfaction of the goal the variant should achieve.

In Table 1b, we show refinement assumptions monitoring for the goals in Fig. 2. In line with the characterization of requirements we gave in Fig. 1, every goal has a target context that interprets its satisfaction criteria concretely. Monitoring refinement assumptions means monitoring if the target contexts for root and intermediate goals are reached or not in each operation. In the table, the first column is an identifier for each operation, whereas the following columns reflect the satisfaction of the goals.

4.2 Autonomic Evolution of Adoptability Assumptions

Traditionally, software selects the variant to achieve its current goals based on the policies defined by its designers. However, the variant the system would choose according to such policies might include adoptability assumptions that the operation experience proved to be invalid. When this is the case, the system can adjust its behaviour autonomically—without human intervention—and choose an alternative variant that contains adoptability assumptions proven more valid in the current context. In a contextual goal model, suppose that the root goal is activated (and some subgoals as well),

and there exist more than one adoptable goal model variant (hereafter *GMV*) for meeting the root goal. In autonomous evolution, the selection between *GMV*s is based on the operation experience the system has. The system will use such experience (exemplified in Table 1a) and select the *GMV* that demonstrated to be the most successful.

In the following, the decision taken by the system is based on the history of each GMV_i in each of its required context variants ($RCV_{i.j}$). Each pair $\langle GMV_i, RCV_{i.j}\rangle$ defines one adoptability assumption saying that variant GMV_i is a valid means to achieve the root goal if the required context $RCV_{i.j}$ holds. We exploit now simple statistical functions to define basic metrics that can be used by a system to select the best GMV based on the operation experience the system has:

- **Assumption Validity** (AV): this factor represents the statistical evidence concerning the capability of GMV_i to reach the root goal when a specific required context $RCV_{i.j}$ holds. Assumption validity uses the monitored data collected in Table 1a. Suppose GMV_i was enacted m times, out of which required context $RCV_{i.j}$ was true n times, out of which the GMV_i led to reach the root goal o times. AV is computed as the ration between the successful executions of GMV_i over all executions in which $RCV_{i.j}$ was true: $AV(GMV_i, RCV_{i.j}) = o/n$
- **Assumption Criticality** (AC): this metric represents the extent to which the falsity of a required context variant $RCV_{i.j}$ prevents GMV_i from achieving the root goal (though some other $RCV_{i.k}$ holds). Suppose GMV_i was enacted p times, out of which $RCV_{i.j}$ was false q times, out of which the $GMV_{i.j}$ did not lead to reach the root goal r times, then $AC(GMV_i, RCV_{i.j}) = r/q$

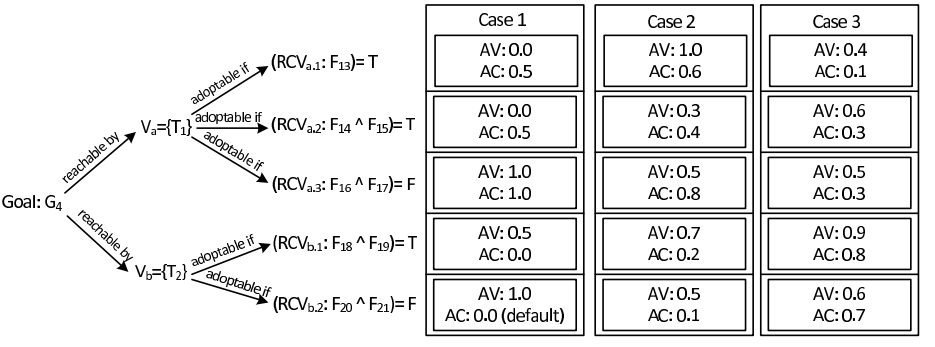


Fig. 3. Three different autonomic evolution scenarios involving the computation of AV and AC

Fig. 3 shows that currently, $RCV_{a.1}$, $RCV_{a.2}$, and $RCV_{b.1}$ hold, whereas $RCV_{a.3}$ and $RCV_{b.2}$ do not hold. It also shows three different cases for the values of AV and AC. The first (Case 1) is computed on the basis of the operation history shown in Table 1a, whereas the other two reflect other operation histories. Upon that, the system will select the goal model variant that is the most likely to reach the root goal. Such likelihood is determined by considering both AV and AC of each pair $\langle GMV_i, RCV_{i.j}\rangle$. Instead of giving one algorithm to elect the GMV to enact, we here outline several

policies that the designer could adopt, and probably change over time, that guide the decision making algorithm.

- **Optimistic:** this policy selects a GMV to enact on the basis of on the holding $RCVs$ having the highest AV metric. This policy is optimistic because it ignores that the very same GMV_i might currently have a false $RCV_{i.j}$ with a high $AC(GMV_i, RCV_{i.j})$ factor. In other words, this policy gives more importance to the positive evidence from the holding $RCVs$ than the negative impact from not-holding ones on the satisfaction of goals. For example, in Case 1 of Fig. 3, the selected variant will be V_b, given that its required context variant $RCV_{b.1}$ is the holding context having maximum AV value (0.5).
- **Sceptical:** this policy selects the goal model variant based on the lowest AC metric. This policy is sceptical because, irrespective of the likelihood of success of a GMV, it will choose a variant that, in the given context, is less likely to fail. The policy gives more importance to the negative impact the false $RCVs$ have on a GMV that the positive evidence the true $RCVs$ give. For example, in Case 2 of Fig. 3, the selected variant will be V_b, because its required context variant $RCV_{b.2}$ is that, among not-holding ones, having lowest AC value (0.1).
- **Balanced:** the selection according to this policy considers both the AV and the AC metrics. It is often the case that, considering AV and AC alone, the selected variant would be different. This is true, for instance, in Case 3 of Fig. 3, where the optimistic policy would choose V_b (due to the high $AV(V_a, RCV_{b.1})$ value which is 0.9), while the sceptical policy would choose V_a (due to the low $AC(V_a, RCV_{a.3})$ value which is 0.3). The balanced policy gives different weights to the two metrics, for instance 50% each. So, the balanced view will choose V_a, due to the $AV(V_a, RCV_{a.2})$ value which is 0.6 and $AC(V_a, RCV_{a.3})$ value which is 0.3.

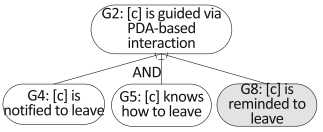
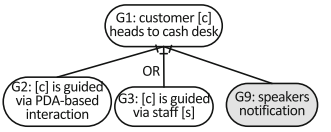
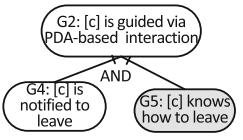
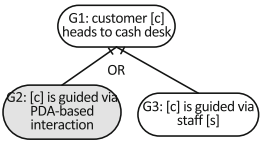
4.3 Designer-Supported Evolution of Refinement Assumptions

Many types of evolutionary actions concerning requirements models cannot be taken autonomously by software. This happens when evolution requires to apply substantial changes in the model, changes that are more radical than updating the rank of software variants which we described in Sec. 4.2. We focus here on evolutionary actions a designer can carry out on the basis of the operation experience history the system has gathered at runtime (Table 1b). We outline two primitive types of evolutionary actions that apply to requirements refinement (decomposition and specialization) assumptions, and explain when these actions should be applied. We illustrate them with the aid of the examples in Table 2.

Add sub-requirements: the refinement of a requirement is changed by adding a new sub-requirement. In a contextual goal model, a new sub-goal is added either to an AND-decomposition or to an OR-decomposition. The first case means that an additional sub-goal should be achieved to reach the parent goal. The second case corresponds to adding a new option to achieve the parent goal:

- Additions to AND-decompositions are needed when the operation history shows that, in many software operations, the achievement of the subgoals was not enough

Table 2. Designer-supported evolution illustrated on the refinement of goal G_2 in Fig. 2

Add to decomposition. G_8 is added when operations like J_3 (Table 1b) occur often. 	**Add to specialization.** G_9 is added when operations like J_4 and J_5 occur often. 
Remove from decomposition. G_5 is removed when operations like J_7 rarely occur, while operations like J_6 occur often. 	**Remove from specialization.** G_2 is removed when operations like J_1 rarely occur, while operations like J_4 occur often. 

to achieve the parent goal. For example, operation J_3 means that a customer was successfully notified (G_4 was reached), he was informed about the way to leave (G_5 was reached), but still he did not leave on time (G_2 and G_1 were not reached). The system is supposed to continuously analyse the operation history (Table 1b) and, if operations like J_3 occur often, then it will ask the designer to take an addition evolutionary action. The designer could add a subgoal like "customer is reminded to leave" (G_8).

– Additions to OR-decompositions are required if the operation history shows that the achievement of alternative sub-goals, even after autonomous evolution, is typically not sufficient to reach the parent goal. For example, operation J_4 means that a customer was successfully guided by his PDA (he read the notification to leave and instructions about the way to leave, reaching therefore G_2), but he did not eventually leave on time (G_1 was not reached). J_5 represents a similar experience with respect to G_3. If operations like J_4 and J_5 occur often, then the system informs the designer asking him to add a new alternative. The designer could add a sub-goal such as "make announcement via the shopping mall public speakers" (G_9).

Remove sub-requirements: the refinement of a requirement is modified by removing a sub-requirement. In an AND-decomposition, a sub-goal is removed, meaning that to achieve the parent goal fewer sub-goals have to be achieved. In an OR-decomposition, removing a sub-goal means deleting an alternative way to achieve the parent goal.

– Removing a sub-goal from an AND-decomposition is applied when that sub-goal is typically unnecessary for the satisfaction of the parent goal. For example, in software operation J_6 the customer was successfully notified (G_4 was reached) but he did not read/receive instructions to leave (G_5 was not met), and still he moved towards the cash desk (G_2 and G_1 were reached). If the parent goal is often satisfied without G_5 being satisfied, the system will inform the designer suggesting to remove it. This will imply removing or disabling all software variants that support such goal from the implemented system.

– Removing a sub-goal from an OR-decomposition is applied when that sub-goal does not usually lead to the satisfaction of its parent goal. For example, if operations like J_4 (explained above) happen very often, this means that notifying customers and leading them by PDAs is an inapplicable alternative (for the root goal is not met) that need not be supported. Thus, the system will suggest the designer to remove this alternative and the corresponding software functionalities.

5 Conclusions and Future Work

Requirements evolution is a main driver for software evolution. Requirements evolve due to many reasons. So far, literature focused mainly on changes in stakeholders' needs. Here, we advocated for and illustrated another main reason for requirements evolution; the assumptions in a requirements model. Assumptions validity is not predictable at design time and, moreover, changes over time. We conceive evolution as a lifelong process that moves software towards a behaviour based on the assumptions proven more valid. Evolution is desirably enacted by the system itself as an autonomic activity. However, certain kinds of evolution are not possible autonomously, and in this case the system can only announce problematic assumptions to designers, asking them to take an appropriate evolutionary action. To support these evolutions, software should monitor runtime operation and diagnose if the assumptions hold. We illustrated our view using contextual goal models as requirements models.

Future work involves mainly three threads. First, we will develop and implement algorithms that enact the principles we introduced in this paper and enable contextual requirements evolution. A crucial role will be played by the decision-making algorithm to evolve the rank of goal model variants. This should be a multi-factor algorithm that considers different dimensions such as operation history, qualities, preferences, timeliness, etc. Second, we will devise and investigate principles to adopt, compose, and switch between policies to select variants. Third, we will define automated reasoning techniques to identify the evolutionary actions suggested to designers and to select which is the best set of evolutionary actions that maximizes positive impact and minimizes costs.

Acknowledgments. This work has been partially funded by the EU Commission, through the ANIKETOS, FastFix, SecureChange, and NESSOS projects and by Science Foundation Ireland grant $03/CE2/I303\_1$.

References

1. Lam, W., Loomes, M.: Requirements evolution in the midst of environmental change: a managed approach. In: Proceedings of CSMR 1998, pp. 121–127 (1998)
2. Harker, S.D.P., Eason, K.D., Dobson, J.E.: The change and evolution of requirements as a challenge to the practice of software engineering. In: Proceedings of RE 2003, pp. 266–272 (1993)
3. Finkelstein, A., Savigni, A.: A framework for requirements engineering for context-aware services. In: Proceedings of STRAW 2001 (2001)
4. Salifu, M., Yu, Y., Nuseibeh, B.: Specifying monitoring and switching problems in context. In: Proceedings of RE 2007, pp. 211–220 (2007)

5. Ali, R., Dalpiaz, F., Giorgini, P.: A goal modeling framework for self-contextualizable software. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) Enterprise, Business-Process and Information Systems Modeling. Lecture Notes in Business Information Processing, vol. 29, pp. 326–338. Springer, Heidelberg (2009)
6. Hartmann, H., Trew, T.: Using feature diagrams with context variability to model multiple product lines for software supply chains. In: Proceedings of SPLC 2008, pp. 12–21 (2008)
7. Ali, R., Dalpiaz, F., Giorgini, P.: A goal-based framework for contextual requirements modeling and analysis. Requirements Engineering 15, 439–458 (2010)
8. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems 8(3), 203–236 (2004)
9. Yu, E.: Modelling strategic relationships for process reengineering. Ph.D. Thesis, University of Toronto (1995)

A Formal Modeling Approach to Information Systems Evolution and Data Migration

Mohammed A. Aboulsamh and Jim Davies

Department of Computer Science
University of Oxford
Oxford UK
{Mohammed.Aboulsamh,Jim.Davies}@comlab.ox.ac.uk

Abstract. In the model-driven approach to software development, system implementations are generated automatically from abstract models of structure and behaviour. This could greatly facilitate systems evolution: a new version of a system may be produced simply by updating the system model and repeating the generation process. However, an information system may hold data of considerable value and complexity, and this must be safely migrated at each evolutionary step. This paper shows how this problem can be solved through a formal, model-driven approach: platform-specific data migration functions are generated automatically from a formal model of system changes, and the applicability of these functions is calculated in advance, ensuring that they may be safely applied to existing data.

Keywords: model-driven, information systems, data migration.

1 Introduction

Model-driven development is the automatic generation of software implementations from abstract models. The abstraction is achieved through the identification of development patterns and heuristics—common to systems of a particular class, or within a particular domain—and their incorporation into code generation or compilation technology. The result is an approach in which concise, descriptions of structure and functionality, often accessible to domain experts, become the 'source code' for part of a system.

That such an approach can reduce the cost of development, while also improving software quality, should be self-evident: the amount of manual programming effort is reduced; the 'code' is more easily reviewed by developers and stakeholders alike; once the generation technology is proven, any questions of validation or verification can be addressed at a higher level of abstraction, where answers are easier to obtain. Furthermore, models that are to be used as the basis for code generation will necessarily admit a clear, precise interpretation: formal techniques such as the B-Method [2] may be applied to check properties of a design prior to implementation.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 383–397, 2011.

For most information systems, however, development is not a one-off activity. These systems will be updated, repeatedly, in response to changes in context and requirements. At each update, the data held within the old system must be transferred to the new implementation. If changes have been made to the data model, then careful consideration must be given to the transformation and representation of existing data—which may be of considerable value and importance to the organisation. In such systems, it is necessary to address the challenges not only of initial development, but also of systems evolution.

In this paper, we show how these challenges can be addressed through a formal, model-driven approach. Using the Unified Modeling Language (UML) as an example, we show how a sequence of proposed changes to a system can themselves be represented as a model. We show how such a model may be used as the basis for the automatic generation of a corresponding data migration function, in the standard Structured Query Language (SQL).

We show also how a formal representation of the model in the Abstract Machine Notation (AMN) of the B-Method allows us to check that this function is applicable—that the migrated data would fit within the constraints of the new system. This applicability information can be fed back to the system designer or architect during the modelling activity, allowing them to see immediately how the changes they are proposing would affect the data in the existing system.

Each of our models or representations—the sequence of proposed changes, the SQL implementation, and the AMN representation—conforms to a specific metamodel, each of which conforms in turn to the Meta Object Facility (MOF) metamodelling standard. The first of these metamodels can be constructed as an extension the metamodel for the system modelling language—in this case, this is the metamodel for UML. The formal, model-driven approach proposed in this paper is thus summarised by the diagram of Figure 1.

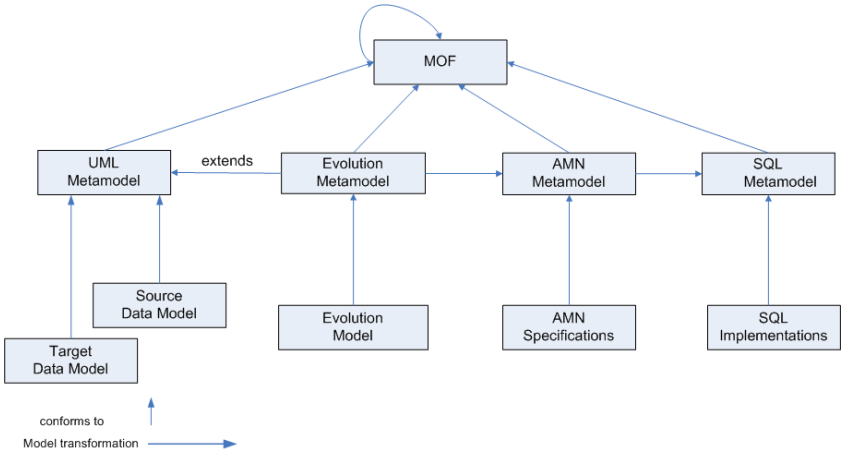


Fig. 1. Overview of the approach

2 Background

Model-Driven Development of Information Systems. The model-driven approach is ideally suited to the development of *information systems*, where the emphasis is upon the management of data rather than the derivation of algorithms. Models have been used for the automatic generation of database components [3], for schema and data integration [4,5], and for data transformation [6]. Although other notations could be used for model-driven development—for example, Entity-Relationship (ER) diagrams [7] and Object Role Models (ORM) [8]—we have chosen to work with models written in UML [9]. UML has been the focus for much of the work on model-driven development, and has a standard, tool-supported notation—the Object Constraint Language [10,11]—for the precise description of constraints for data management.

Formal modeling with B. The B-Method [2] is a formal specification technique with two notations: the Abstract Machine Notation (AMN), used to specify the state space of a system; and the Generalised Substitution Language (GSL), used to specify system operations. A model in AMN comprises one or more abstract machines, each of which is described in a series of clauses, as shown in Figure 2: **SETS** are basic value domains; **VARIABLES** characterise the state of the system, which must satisfy the constraint of the **INVARIANT** clause; **INITIALIZATION** describes the initial state; **OPERATIONS** are described as GSL statements. A GSL statement is a program consisting of basic substitutions connected by choice, sequential, and parallel operators. Of particular interest is the preconditioning construct $P \mid S$, whose behaviour is that of S if condition P is true, and is otherwise undefined; $@x.(P ==> S)$ represents substitution S parameterised by an arbitrary x satisfying P: see Figure 2. Properties of models

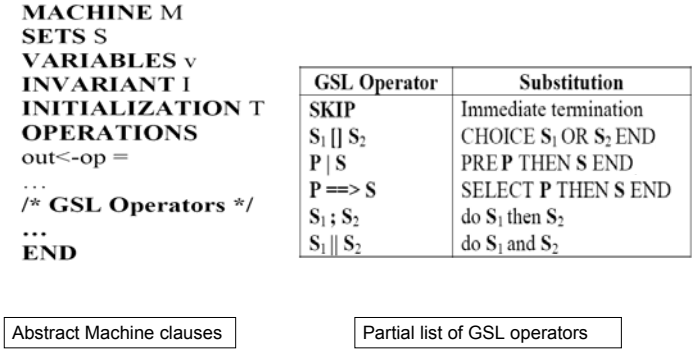


Fig. 2. Main elements of B-method AMN and GSL notations

written in AMN and GSL can be verified, often automatically, using a range of proof engines: see, for example, [14].

Running Example. We will use a simple UML model of an employee database to illustrate our approach. Figure 3 represents an initial version, consisting of classes Employee, Department, and PersonalInfo, together with a number of associations between these classes—one of which, the manager association, is optional—and a number of OCL constraints.

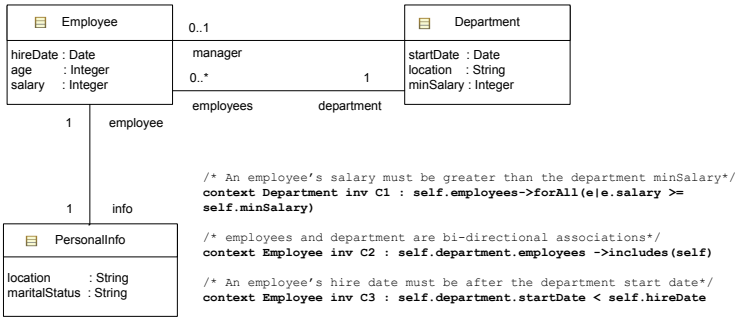


Fig. 3. Data model of a simplified Employee Information System

A valid instance of this model is shown in Figure 4. In a second version, introduced later, we will imagine that the provision of a separate PersonalInfo class has been deemed unnecessary, that the manager relationship is now mandatory, and that a more detailed account of employee assignment is required.

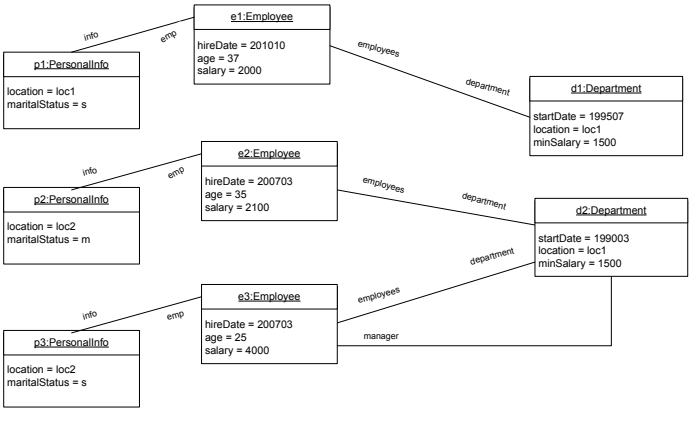


Fig. 4. Object models of a simplified Employee Information System

3 Modelling Evolution

3.1 Abstract Syntax

In updating a model to reflect a proposed system evolution, we may add, modify, or delete model elements. In the case of UML, the range of elements that may be affected is described by the UML metamodel—the language definition. Our language of changes will thus include basic statements such as `addClass()` or `modifyAssociation()`, as well as derived terms corresponding to operations allowed by a specific model editor.

To model evolutionary steps involving changes to more than one model element, we define *combinators* for our language, allowing us to compose changes both sequentially (`;`) and in parallel (`||`). The second of these is essential if we wish to achieve a properly abstract description of updates where sequential ordering would require the introduction of temporary variables, such as an exchange of roles between two different properties, or where the ordering is unimportant, as long as the updates are performed as a transaction upon the model data.

Figure 5 shows the main concepts of our proposed metamodel for evolution, along with related (shaded) classes drawn from MOF and OCL. An instance of the Evolution class is a description of the relationship between source and target models, and this class is presented as a subclass of both Package and Class: as a package, it provides a namespace for EvolutionElements; as a class, it can define properties and operations. An instance of Evolution can use OclExpressions to describe invariant properties and well-formedness rules; it contains also a set of EvolutionElements, which may be specialised to address different

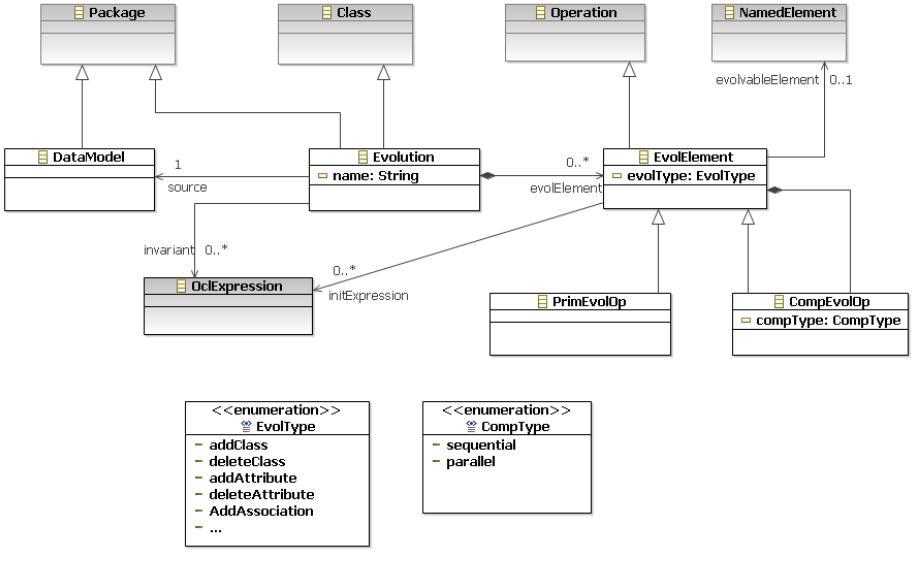


Fig. 5. Evolution Metamodel

types of changes. A change could be primitive, corresponding to a single model element: for example, `addClass()` and `deleteAssociation()`. Alternatively, it could be an instance of CompoundEvolutionOperation, corresponding to multiple elements: for example, `extractClass()`. This enumeration could be extended to reflect particular aspects of the development domain, or particular features of the modelling toolset.

An EvolutionElement instance has a set of typed parameters: for deletion operations, these parameters will identify the class and property concerned; for additions or modifications, they may represent the type or initial value, described as an `OclExpression`, for example,

```
addAttribute(cName, name, type : String, exp : OclExpression)
```

A CompoundEvolutionOperation may represent a model evolution pattern. For example, the inlining of a redundant class, in which a class is removed from the model and its properties transferred to an associated class, may be specified as

```
inlineClass(srcClass, tgtClass, srcAttribute : String)
```

where `srcClass` is the class receiving the properties, `tgtClass` is the class to be deleted, and `srcAttribute` is the association relationship between them. Other commonly-used patterns include the extraction of a class and the movement of attributes up and down inheritance hierarchies. A wide range of patterns have been identified in the literature: see for example, [15], [16], and [17].

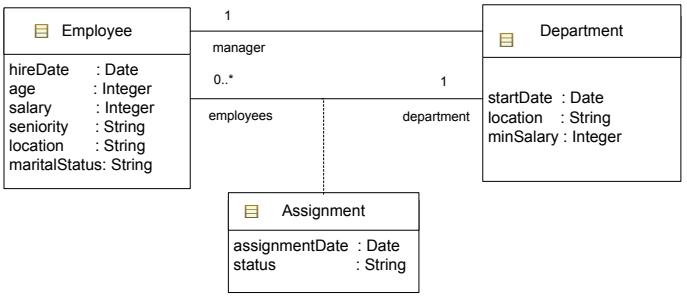
Example. Consider the evolutionary change to our employee information system outlined at the end of the previous section. This may be described as follows:

```
inlineClass(Employee, employee, PersonalInfo);
addAttribute(Employee, seniority: String
   [if self.age > 50 then 'senior' else 'junior']) ;
addAssociationClass(Employee, department, Department,
   employees, Assignment);
addAttribute(Assignment,assignmentDate:Date
   [self.employees.hireDate=self.assignmentDate)]) ;
addAttribute(Assignment, status: String [self.status = 'active']) ;
modifyAssociation(Department, manager, Employee, 1,1) ;
```

Figure 6 shows the evolved version of the data model after the above evolution operations have been performed.

3.2 Formal Semantics

To reason about a proposed evolution, and determine its applicability, we need to determine the constraint information in the target (evolved) model; this requires the definition of an appropriate constraint semantics for the system modelling language: in this case, for the UML. We may do this using the AMN language of the B-Method, characterising model constraints as abstract machine




```
context Department inv C1 : self.employees->forAll(e|e.salary >= self.minSalary)
context Employee   inv C2 : self.department.employees->includes(self)
context Employee   inv C3 : self.department.startDate <  self.hireDate
```

Fig. 6. Employee Information System model (evolved)

specifications. We need also to represent the intended effect of the evolution; this requires a complementary semantics for our evolution language. We may do this using the GSL notation of the B-Method.

UML Constraint Semantics. In a similar fashion to [22], we may map UML data model into an AMN machine by translating UML metamodel concepts into AMN machine structures. Our mapping may be characterised as follows:

```
MACHINE
  DataModel
SETS
  CLASS; ObjectID; ATTRIBUTE; ASSOCIATION; VALUE; TYPE
VARIABLES
  class, attribute, association, value, link
INVARIANT
  class       : CLASS +-> POW(ObjectID)
  attribute   : CLASS +-> ATTRIBUTE   +-> TYPE
  value       : CLASS +-> ATTRIBUTE   +-> ObjectID +-> VALUE
  association: CLASS +-> ASSOCIATION +-> CLASS
  link        : CLASS +-> ASSOCIATION +-> ObjectID +-> POW(ObjectID)
```

The `SETS` clause introduces the sets of possible names for classes, attributes, associations, as well as: the set `ObjectID` of possible object references; the set `Type` of possible attribute or parameter types; and the set `Value` of possible values of attributes.

The `INVARIANT` clause declares: a function `class` to map a class name to a set of `ObjectID`s, representing its current extent (`POW` is the powerset operator in AMN); a function `attribute` to map each attribute, identified first by the name of the class, and then by the name of the attribute itself, to a corresponding type; a function `association`, as a functional relation between class names; a function `value`, representing the current value of the named property for each object; a function `link`, as a relation between object identifiers.

Example. An instantiation of the above machine to match the object model of Figures 3 and 4 can be described by the following relations:

```
class       = {(Employee   |-> {e1, e2, e3}),... }
attribute   = {(Employee   |-> salary    |-> INTEGER),...}
association = {(Employee   |-> department|-> Department),...}
value       = {(Employee   |-> salary    |-> e1 |-> 2000),... }
link        = {(Department |-> employees |-> d1 |-> {e1}) ,...}
```

where `a |-> b` is the AMN pair notation.

Model constraints that require representation of data instances are mapped into invariant properties constraining the `class`, `attribute`, `association`, `value` and `link` variables.

Example. In our system model, the OCL constraint

```
context Employee inv C1 :
  self.department.employees->includes(self)
```

can be mapped to the `INVARIANT`

```
! ee : class (Employee).
     ! dd : link (Employee) (department) (ee) .
        ee : link (Department) (employees) (dd)
```

where `!` denotes universal quantification, `:` denotes a declaration of set membership, and `( )` denotes function application.

Semantics of Evolution Operations. We may give semantics to our evolution operations by mapping each operation to a substitution on the variables of the machine state `class`, `attribute`, `association`, `value`, `link`. These substitutions may be composed, sequentially or in parallel, to produce a specification of the data migration corresponding to a compound, evolutionary change.

For example, the operation `addClass(name)` is mapped to an AMN operation with the following GSL substitution:

```
name /: dom(class) ==>
   (class := class \/ { name |-> {} }
      || value := value \/ { name |-> {} }
      || link  := link  \/ { name |-> {} } )
```

The guard operator `==>` has been used to indicate that this operation is applicable only when `name` is not already present in the domain of function `class`: that is, when the proposed class name is not currently in use for this system. The parallel assignments (denoted by `||`) update the variables `class`, `value`, and `link`. Using set union (denoted by `\/`), we map the new class `name` to empty set of objects, attribute values, and links, respectively.

The operation `deleteClass(name)` maps to a more complex substitution:

```
name : dom(class) ==>
   ( ! o : class(name) .
     ! c : dom(class) - {name} .
     ! n : dom(link(c)) .
     ! p : dom(link(c)(n)) .
       link(c)(n)(p) := link(c)(n)(p) - {o}
     ;
         class   := { name } <<| class
       || value  := { name } <<| value
       || link   := { name } <<| link )
```

This operation consists in two phases: first, we remove any references to objects of class name; then we remove all objects, attribute values, and links owned by that class. The symbol `<<|` denotes domain subtraction: any objects corresponding to `name` are deleted from the mapping `class`; any attributes of, or links originating from, these objects are deleted from `value` and `link`, respectively.

The introduction of a new attribute or association requires an additional parameter: an expression in OCL. This describes an initial set of values, or an initial set of links, respectively. To map these operations, we translate OCL expressions into AMN expressions and choose an appropriate instantiation for the variable `self`, as a reference to the current object. For example, the operation `addAttribute(cName,name,type,exp)` is mapped to:

```
name /: dom(value(cName)) ==>
! o : class(cName) .
   attribute := attribute \/ { cName |-> name |-> type }
    || value := value     \/ { cName |-> name |-> o |->
                   translate(exp)[self  := o] }
```

For each object `o` of the specified class `cName`, the new attribute `name` is associated with the value of the expression `exp`, with `o` substituted for `self`.

The operations of modifying existing classes, attributes or associations, require a similar semantic treatment. The only differences are that: the modifications to `value`, `link` and `operation` are represented using functional overriding: the current values of the attributes are replaced with the values of `exp`, suitably instantiated; the condition upon `name` is negated.

To give a semantics to complex operations and evolution patterns, we may expand the substitution obtained from the sequential or parallel combinators, or produce a direct definition using raw GSL. For example, the `inlineClass()` operation can be mapped to a two-phase substitution. The first phase moves all of the attributes of the target class, along with their current values, into the source class. The second phase then deletes all objects of the target class, and then the class itself, along with the corresponding sets of attributes and links. For the arguments `srcClass`, `tgtClass`, and `srcAttribute`, this operation has the following semantics:

```
( srcClass : dom(class) &
  tgtClass : dom(class) &
  srcAttribute : dom(link(srcClass)) &
  ! o : class(srcClass) .
     card(link(srcClass)(srcAttribute)(o)) <= 1)
  ==>
   !o : class(srcClass) .
      ! p : link(srcClass)(srcAttribute)(o) .
         ! n : dom(value(tgtClass)) .
            value := value \/ { srcClass |-> n |-> o |->
                     value(tgtClass)(p)(n) }
  ;
  [[deleteClass(tgtClass)]]
```

where `[[deleteClass(tgtClass)]]` denotes the semantics of the `deleteClass` operation defined earlier.

Example. The AMN semantics of the proposed evolution of our employee information system can be obtained by instantiating the functions presented above. For example, the semantics of the evolution operation `inlineClass()` would be instantiated to produce:

```
inlineClass(Employee, employee, PersonalInfo) =
( Employee : dom(class) &
  PersonalInfo : dom(class) &
  employee : dom(link(Employee)) &
  ! o : class(Employee) .
     card(link(Employee)(employee)(o)) <= 1)
  ==>
   !o : class(Employee) .
      ! p : link(Employee)(employee)(o) .
         ! n : dom(value(PersonalInfo)) .
            value := value \/ { Employee |-> n |-> o |->
                     value(PersonalInfo)(p)(n) }
  ;
  [[deleteClass(PersonalInfo)]]
```

Note that some evolution operations will have implications for data integrity and model conformance. For example, the `modifyAssociation()` operation in section 3.1 changes the multiplicity of the *manager* association from optional to mandatory. Accordingly, we would expect a non-trivial guard of the form

```
! dd : class (Department) .
   link (Department ) (manager) (dd) /= { }
```

to appear in the applicability constraint for our generated data migration function, where `/=` and `{}` denote set inequality and the empty set, respectively. This guard requires that the manager link should already exist in all objects of class

Department. This guard is satisfied by the Department instance d2 in Figure 4 which has a manager link to Employee instance e3. However, Department instance d1 does not have such a link and may not be migrated to the new model until such a link is established. This can be achieved by providing additional input: a set of links or references to be created for the association in question—or an expression explaining how these links are to be created.

4 Generating Data Migrations

To generate an appropriate implementation of our model evolution and data transformation, we require a mapping from our abstract model operations to operations upon a specific, concrete platform: some of these operations will update metadata, or features of the representation (such as tables in a database); others will be data operations implementing the semantics outlined above.

In practice, the kind of data that we might wish to preserve across successive versions of an information system is likely to be stored in a relational database. In model-driven development, the schema for such a database will have been derived automatically from the data model. For the purposes of this paper, we assume a simple derivation, in which each class or association is represented as a separate table, with rows corresponding to objects or links, respectively, and each value attribute is represented as a column. Any computable mapping could be adopted for naming tables and columns; here, we will assume that tables and columns representing classes and attributes will take the name of class or attribute in question, and that tables representing associations will be named by concatenating the name of the class and the linking association property. We will write `<a_b>` to denote the concatenation of two names a and b.

Having chosen a particular derivation, we may define a second interpretation for our model evolution operations. The operation `addClass(name)`, for example, corresponds to the following SQL statement:

```
CREATE TABLE <name> (
<name_id> INT NOT NULL,
PRIMARY KEY (<name_id>)
);
```

As another example of SQL interpretation of a primitive model evolution operation, consider the `addAttribute (cName, name, type, exp)` operation. This operation can be implemented as a pair of updates: first to the schema, and then to the rows of the table in question. Assuming a simple mapping from `type` to SQL primitive types, this operation can be given the following interpretation in SQL:

```
ALTER TABLE <class>
ADD COLUMN (<name> type)
UPDATE <class> SET [[name = exp]]
```

where [[name = exp]] represents the SQL implementation of the specified assignment, instantiated with suitable object identifiers.

The compound evolution operations such as inlineClass can be implemented as a sequence of addAttribute operations, one for each of the attributes of the target class, followed by an update to insert the appropriate values, then a delete to target class:

```
ALTER TABLE (<srcClass>),
ADD COLUMN   <srcClass_tgtClassAttribute>
UPDATE       <srcClass>
SET          <srcClass> . <tgtClassAttribute> =
(SELECT <tgtClass> . <tgtClassAttribute>
 FROM   <tgtClass>
 WHERE  <tgtClass> . <srchAttribute> = <srcClass_id>);

DROP TABLE < tgtClass>;
```

Example. Using the SQL mapping rules outlined above, the inlineClass () operation, used in the evolve the Employee Assignment Tracking System model, can be given the following interpretation in SQL:

```
ALTER TABLE Employee,
ADD COLUMN   location

UPDATE       Employee
SET          Employee . location =
(SELECT PersonalInfo . location
 FROM   PersonalInfo
 WHERE  PersonalInfo . employee = employee_id);

DROP TABLE < PersonalInfo>;
```

5 Discussion

The model-driven approach offers an opportunity to consider the question of information system evolution and data migration, in detail, at the design stage. In this paper we have outlined a possible solution: capturing the changes to a model using a language of model operations, mapping each operation to a formal specification of the data model and corresponding data transformation, checking a sequence of operations for consistency with respect to the model semantics and the existing data, and—for a specific platform—automating the process of implementation.

We use a formal semantics for the description of evolution, and for the data model to generate a correctness condition, as a test to be performed before the migration takes place. If the correctness condition is satisfied, then the generated migration program is guaranteed to succeed: the transformed data will satisfy

the integrity constraints of the new system. Not to do so would mean that we ran the risk of our automatic migration doing exactly what the specification says, but leaving the data in an unacceptable state.

We have refined our approach to information systems evolution and data migration based on work that has been going in our research group for few years. In [25], the idea of using precise object models as a basis for generating working database implementation was investigated: an object modeling notation was introduced in which all operations are described purely in terms of their intended effect upon data, using a formal notation based upon AMN. In [23] and [24], we outlined the main concepts of the approach we detailed here: by showing how changes to an object model can be reflected on the structure of the model and on the representation of the data stored—without, however, the description of the formal semantics and mapping to SQL presented here.

The work we describe in this paper relates to the intersection of two main research areas: database schema evolution and Model-Driven Engineering (MDE). Schema evolution is the process of applying changes to a schema in a consistent way and propagating these changes to the instances while the database is in operation [15]. Schema evolution has been widely discussed in literature and therefore, various approaches have been proposed. Some of the most relevant approaches to the general problem of information system evolution are [15], [16], and [17]. While these and other attempts provide solid theoretical foundations and interesting methodological approach, the lack of abstraction was observed in [18] and remains largely unsolved after many years.

Advocates of Model-Driven Engineering have promoted the idea of abstracting from implementation details by focusing on models as first class entities. The abstraction answer to the issue of information system evolution we are addressing here builds upon some of the most recent results on model-driven engineering literature such as Model transformation [26]. Model weaving [19] and Model refactoring [20]. These and other approaches in MDE may help in characterizing information systems evolution however, they remain largely general-purpose and offer no specific support for information systems evolution tasks such as data migration.

In this paper we have addressed information systems evolution and data migration problem from a Model-Driven Engineering perspective. We have shown that any evolutionary step has potential consequences for existing data. Manual data migration, in the face of complex integrity constraints and business rules, can be an expensive task. In addition, considering evolutionary changes and consequent data transformation at an implementation level can be complex and error-prone.

Our approach may be extended to address not only the consistency of data, but also the applicability of particular operations or workflows. If an operation or workflow is associated with a particular precondition, written in OCL, then we may map this to an additional constraint in AMN, and check to see whether its validity would be affected by the proposed data transformation.

References

1. Bezivin, J.: On the unification power of models. Software and Systems Modeling 4(2), 171–188 (2005)
2. Abrial, J.R.: The B-book: Assigning Programs to Meanings. Cambridge University Press, Cambridge (1996)
3. Davies, J., Welch, J., Cavarra, A., Crichton, E.: On the Generation of Object Databases using Booster. In: The 11th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2006), pp. 249–258. IEEE Computer Society, Washington, DC, USA (2006)
4. Kurz, S., Guppenberger, M., Freitag, B.: A UML profile for modeling schema mappings. In: Roddick, J., Benjamins, V.R., Si-said Cherfi, S., Chiang, R., Claramunt, C., Elmasri, R.A., Grandi, F., Han, H., Hepp, M., Lytras, M.D., Mišić, V.B., Poels, G., Song, I.-Y., Trujillo, J., Vangenot, C. (eds.) ER Workshops 2006. LNCS, vol. 4231, pp. 53–62. Springer, Heidelberg (2006)
5. Philippi, S.: Model driven generation and testing of object-relational mappings. Journal of Systems and Software 77(2), 193–207 (2005)
6. Gogolla, M., Lindow, A.: Transforming data models with UML. In: Knowledge Transformation for the Semantic Web, pp. 18–33 (2003)
7. Thalheim, B.: Fundamentals of Entity-relationship Modeling. Springer, Heidelberg (1999)
8. Halpin, T., Morgan, T.: Information Modeling and Relational Databases. Morgan Kaufmann Publishers Inc., San Francisco (2008)
9. Object Management Group(OMG). UML 2.0 infrastructure specification, v2.1.2 (2007), http://www.omg.org/spec/UML/2.1.2/ (retrieved February 09, 2011)
10. Object Management Group (OMG). OCL specifications, version 2.2. (2010), http://www.omg.org/spec/OCL/2.2 (retrieved February 09, 2011)
11. Demuth, B., Hussmann, H.: Using UML/OCL constraints for relational database design. In: France, R.B. (ed.) UML 1999. LNCS, vol. 1723, pp. 598–613. Springer, Heidelberg (1999)
12. Gries, D.: The Science of Programming. Springer, New York (1981)
13. Dijkstra, E.W.: A Discipline of Programming. Prentice-Hall, Englewood Cliffs (1976)
14. B-Core UK. B-Toolkit (2011), http://www.b-core.com/btoolkit.html (retrieved February 09, 2011)
15. Banerjee, J., Kim, W., Kim, H.-J., Korth, H.: Semantics and implementation of schema evolution in object-oriented databases. In: ACM SIGMOD International Conference on Management of Data (SIGMOD 1987), pp. 311–322. ACM, New York (1987)
16. Ferrandina, F., Meyer, T., Zicari, R., Ferran, G., Madec, J.: Schema and database evolution in the O2 object database system. In: Very Large Database, Morgan Kaufmann, San Francisco (1995)
17. Jing, J., Claypool, K., Rundensteiner, E.: SERF: Schema Evolution through an Extensible, Re-usable and Flexible Framework. In: Int. Conf. on Information and Knowledge Management (1998)
18. Rashid, A., Sawyer, P., Pulvermueller, E.: A flexible approach for instance adaptation during class versioning. In: Proceedings of the International Symposium on Objects and Databases, pp. 101–113. Springer, London (2000)
19. Fabro, M., Bezivin, J., Jouault, F., Breton, E., Gueltas, G.A.: a generic model weaver. In: Proceedings of the 1re Journe sur l'Ingnierie Dirige par les Modles (2005)

20. Sunye, G., Pollet, D., Traon, Y., Jezequel, J.-M.: Refactoring UML models. In: The 4th International Conference on The Modeling Languages, pp. 134–148. Springer, Heidelberg (2001)
21. Peng, S.-L., Clark, D., Androutsopoulos, K.: UML to B: Formal verification of object-oriented models. In: Boiten, E.A., Derrick, J., Smith, G.P. (eds.) IFM 2004. LNCS, vol. 2999, pp. 187–206. Springer, Heidelberg (2004)
22. Laleau, R., Mammar, A.: An Overview of a Method and Its Support Tool for Generating B Specifications from UML Notations. In: IEEE Proceedings of the Automated Software Engineering, pp. 269–272 (2000)
23. Aboulsamh, M., Crichton, E., Davies, J., Welch, J.: Model-driven data migration. In: 2010 International Conference on Advances in Conceptual Modeling: Applications and Challenges (ER 2010), pp. 285–294. Springer, Heidelberg (2010)
24. Aboulsamh, M., Davies, J.: A Metamodel-Based Approach to Information Systems Evolution and Data Migration. In: The 2010 Fifth International Conference on Software Engineering Advances (ICSEA 2010), IEEE Computer Society, Washington, DC, USA (2010b)
25. Davies, J., Crichton, C., Crichton, E., Neilson, D., Sorensen, I.H.: Formality, Evolution, and Model-driven Software Engineering. Electron. Notes Theor. Comput. Sci. (130), 39–55 (2005)
26. Sendall, S., Kozaczynski, W.: Model transformation: The heart and soul of model-driven software development. IEEE Software 20(5), 42–45 (2003)

Overlaying Conceptualizations for Managing Complexity of Scenario Specifications

Remigijus Gustas

Department of Information Systems, Karlstad University, Sweden
Remigijus.Gustas@kau.se

Abstract. Most conventional conceptual modeling approaches are not putting into a foreground interaction dependencies between actors. This is one of the main reasons why it is difficult to apply them for managing complexity of conceptual representations. The goal of this paper is to present conceptual modeling method, which allows constructing graphical representations of scenarios with a more comprehensible structure. Using simple interaction loops between organizational and technical components help designers to separate crosscutting concerns in system engineering without the requirement to specify a complete solution. The examples of sequential, iterative, parallel and alternative behavior are analyzed to demonstrate conceptual descriptions of use-case scenarios. The overlaying and underlying interaction loops among actors are easier to understand, extend and maintain.

Keywords: Interaction dependencies, separation of concerns, conversation for action schema, interaction loops, scenarios.

1 Introduction

Most traditional system analysis and design methods are restricted in their ability to distinguish among crosscutting concerns, which are spanning across various types of diagrams. It does not matter whether designers are using structured analysis and design (SAD) methods [1], [2], component based or object-oriented methods [3]: their expressive power is limited for keeping concerns separate. This situation gives rise to some serious system engineering issues, which remain problematic in the last three decades. Managing complexity of conceptual representations is recognized as one of the difficult problems in the area of system analysis and design. Unfortunately, the foundations of conventional engineering methods are quite weak in addressing it. This is one of the main reasons why the way systems are currently built is very primitive. A more natural and more systematic approach for partitioning is necessary to manage complexity when introducing evolutionary changes of conceptual representations. Grouping of concepts into layers according to their change rate [4] is just a partial solution to the problem.

In the traditional engineering, developers are able to present their design decisions by using a finalized computation neutral representation. This is not a case in the area of system engineering. The limitations of conventional information system modeling

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 398–412, 2011.

approaches result in two side effects, which, in aspect-oriented software development [5], are known as tangling and scattering. Tangling occurs when the software component or class, instead of fulfilling a particular concern, encapsulates a diverse set of concerns. If a particular concern is spread across multiple components, then this situation is called scattering. When the requirements caused by that concern are modified, the designer must identify all related components and to find out how these components are affected by introduced changes. It is especially problematic to perform modification of requirements, which are related to a big number of diagrams. Poor understanding of the natural modularity of concerns makes it difficult to introduce even simple evolutionary extensions of information system (IS) specifications.

Information, decision and resource exchange flows [6] among actors, can be used for keeping crosscutting concerns separate. Nevertheless, most conceptual modeling methods do not put into a foreground modeling interaction flows between organizational and technical components. A starting point in the traditional system development approaches is typically the specification of static dependencies between concepts, which can be represented by relations between various classes of objects. Interaction modeling in terms of Data Flow Diagrams (DFD) was the strength of SAD methods. Unified Modeling language (UML) also supports various types of associations between actors and use cases, but modeling of value and data flows between subsystems is awkward in UML. Aspect-oriented system development is very much in line with the use-case thinking. Nevertheless, there is one fundamental problem with a use-case driven modeling. Use-case diagrams demonstrate decomposition of system functionality and therefore it is difficult to take into account interdependencies between the static and dynamic aspects of a particular concern in a very early modeling phase.

The declarative nature of interaction flows allows analyzing them in the context of events, which are comprehensible for business modeling experts, enterprise architects, system designers and users. Business events can be used as a guidance to move smoothly from system analysis to design, without a requirement to represent a complete solution. For instance, the interaction pattern diagrams [7] are suitable for representing the essential configurations of workflows on different levels of abstraction. This paper demonstrates how simple workflow loops can be used to construct unambiguous graphical descriptions of scenarios with sequential, iterative, synchronized and alternative behavior. Most information system methodologies are quite weak in conceptual modeling of alternative sub-flows and representing consequences if commitments between actors are broken. One of the goals of this paper is illustration of the special kind of interaction dependency, which can be used to preserve the modularity of concerns and to bridge them with the behavioral effects and structural changes in various classes of objects.

The DEMO method provides a solid foundation for interaction based thinking [8]. However, the bridging between the constructs of DEMO process structure diagram and traditional conceptual modeling methods is not totally clear. It is common to all system analysis and design methods to separate disparate views and dimensions of enterprise architecture [9]. The major reason of view separation is that a human limited mind allows focusing on a single aspect in isolation. The traditional IS design methods have insufficient expressive power for modeling coordinating service interactions and value flows. One of the major flaws of the conventional information

system modeling approaches has been unclear integration principles between static and dynamic aspects of conceptualizations. This is because nearly all object-oriented modeling techniques are based on collections [10] of not integrated meta-models. Except OPM [11], the majority of conceptual modeling approaches are plagued by the paradigm mismatch between the diagrammatic constructs for representation of structure and behavior. Although the OPM methodology has an inbuilt zooming mechanism, it lacks a clear idea for keeping crosscutting concerns separate in system analysis and design. Therefore, in this paper we present the conceptual modeling method, which can be used for separation and composition of crosscutting concerns.

2 Using Interaction Loops for Analysis of Crosscutting Concerns

Business process scenarios can be conceptualized by identifying essential flows of events that can be expressed as a set of purposeful interactions between organizational and technical components. Technical components correspond to enterprise subsystems such as machines, software and hardware. Organizational components can be humans, organizations and their divisions or roles, which denote groups of people. Interaction dependencies among actors are important for separation of crosscutting concerns. By walking through interaction dependencies, it is possible to explore various ways in which enterprise system components can be used. In this section, we demonstrate how interaction flows are composed into workflow loops [12]. A workflow loop can be considered as a basic element of scenario, which describes interplay between service requesters and service providers. In its simplest form, a workflow loop is viewed as a response to request that provides a value to a service requester.

We distinguish between actors and passive concepts [7]. An actor can only be represented as an active concept. An instance of an actor is an autonomous subsystem. Its existence can be motivated by a set of interaction dependencies with other actors that keep this subsystem viable. Interaction dependency

$$R(A \cdots \blacktriangleright B)$$

between two active concepts indicates that actor A is an agent. An agent can trigger action R on one or more recipients who are represented by concept B. Actions manipulate objects and their properties that are represented by passive concepts (see next section). The graphical notation of three different types of interaction dependencies (Gustas, 2010) between actors is presented in figure 1.

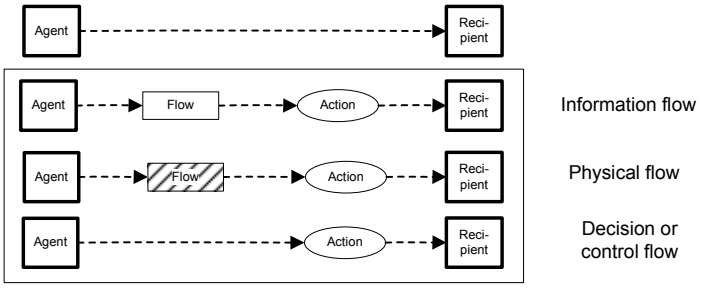


Fig. 1. Three types of interaction dependencies

Interaction dependencies (▪▪▪▪►) are graphically indicated by broken arrows, which denote moving things such as information, decisions or materials. They can be used for keeping track of business events between the actors involved. Actors are represented by square rectangles and actions are represented by ellipses. Solid rectangles denote physical flows and light boxes - data flows. Actions represent legal ways in which various actors are able to interact with each other. Service architectures can be characterized by a number of interaction flows into opposite directions between a service requester and service provider [13], [14].

Service interaction loops are very useful to analyze continuity of value creation process, which captures service value exchange between two or more parties. Both requests and responses are viewed as necessary business events. Such understanding of service interactions is consistent with the ontological foundation of service process. According to Ferrario and Guarino [15], services cannot be transferable, because they are events, not objects. Service providers are actors who receive service requests and transform them into responses, which are sent to service requesters. This idea is illustrated graphically in figure 2.

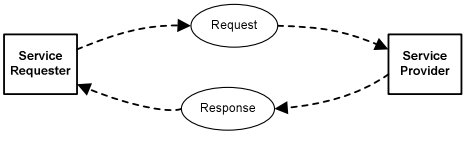


Fig. 2. Basic interaction loop

The presented simple interaction loop can be used in a very early conceptual modeling phase for analysis of crosscutting concerns in terms of communication actions among organizational and technical components. Service responses cannot be delivered without initiating service requests. A response can be viewed in a number of ways. It can be represented by a promise to deliver a desirable result to service requester or it can be viewed as statement, which brings a desired value flow [6] to service requester. Two loosely coupled actors will be represented by the following expression:

If Request(Service Requester ▪▪▪▪►Service Provider)
then Response(Service Provider ▪▪▪▪► Service Requester).

Sequences of interaction events are crucial for analyzing scenarios, which are expressed in terms of requests and responses between actors. For example, Create Reservation action (see figure 3) can be viewed as a promise in connection to Request Room action. Pay action can be understood as a response in exchange to Provide Hotel Room action. Service interaction loops can be delegated to various organizational and technical components. For instance, two interaction loops, which are delegated to different Hotel components such as Reception and Hotel Reservation System, are represented in figure 3.

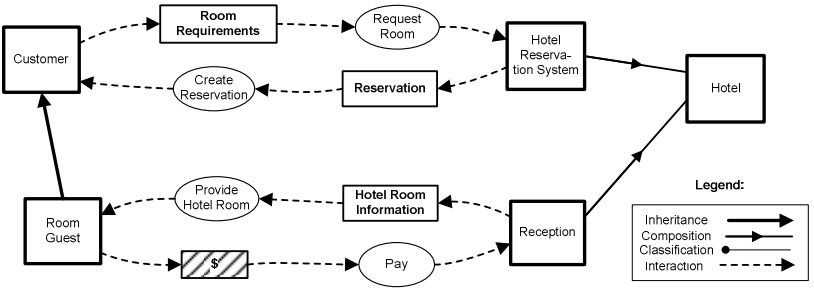


Fig. 3. Two interaction loops in a Hotel Management System

This diagram demonstrates interaction possibilities, which are available to various actors. Actors can be related by inheritance, composition and classification dependencies [13]. These dependencies are used for reasoning about various ways in which interaction loops can be composed, merged or stacked on the top of each other. All outgoing actions represent actor possibilities, which can be interpreted as rights, responsibilities, commitments and claims. A Customer has a right to Request Room by informing a Hotel Reservation System about Room Requirements. If the requested type of room is available, then a Hotel Reservation System has responsibility to Create Reservation for Customer. By taking advantage of the available possibilities, actors may enter into commitments regarding their obligations. For instance, the effect of successfully executed Create Reservation action is a commitment to provide a hotel room for a Customer. Interaction dependencies are inherited by more specific actors and they are propagated to compositional wholes according to the special inference rules [13], [16]. For example, Create Reservation is derived responsibility of a Hotel. If Hotel creates a Reservation, then it is obliged to Provide Hotel Room for the Room Guest. On the other hand, if Hotel is obliged to Provide Hotel Room, then it is entitled to claim a payment. The traditional IS analysis and design methods are not suitable for modeling commitments and claims.

3 Interplay of Interactive and Behavioral Aspects

Interaction dependencies are extensively used in a foreground of enterprise engineering methods [8]. These methods are rooted in the interaction pattern analysis and philosophy of language. The underlying idea of interaction pattern analysis can be explained by using a well-known conversation for action schema [17]. The purpose of introducing this schema was initially motivated by the idea of creating computer based tools for conducting conversations. The goal of this paper is different. We are going to demonstrate how to apply the interaction dependencies in combination with conventional semantic relations, which are used in the area of system analysis and design, for separation of crosscutting concerns. Interaction loops can be expressed by interplay of coordination or production actions, which appear to occur in a particular pattern. This pattern is represented in figure 4.

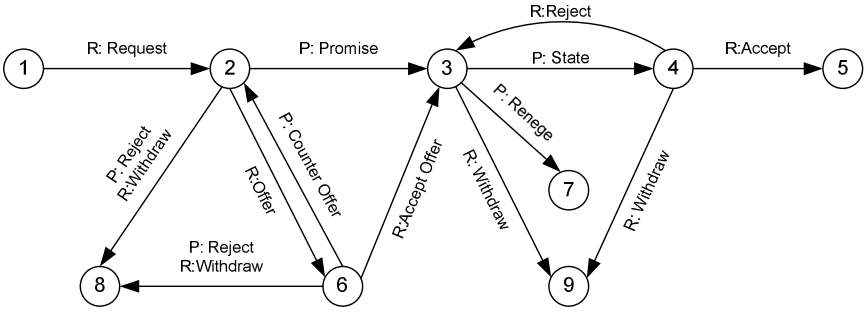


Fig. 4. Conversation for action schema, which is defined by Winograd and Flores

The main idea behind a conversation for action schema can be explained as turn-taking. Any service interaction can be characterized by the same four types of main actions: 1) Request, 2) Promise, 3) State and 4) Accept. Service Requester (R) typically initiates a request (R:Request) and then is waiting for a particular promise (P:Promise) or for a service provision flow (P:State) from Service Provider (P). Request, promise and acceptance are coordination actions. The coordination actions together with the production action (P:State) represent an expected sequence of interactions between service requester and service provider. There are some alternative businesses events such as reject, withdraw, offer, etc. They are necessary for actors to deal with unexpected or undesirable scenarios. The alternative actions must be introduced to handle the breakdowns in the main interaction pattern. For instance, service provider may fail to deliver a promise in time.

The desirable or undesirable communication actions are necessary for triggering various state transitions, which are represented in figure 4. For instance, Create Reservation action in figure 3 can be interpreted as a promise to Provide Hotel Room. Request Room and Create Reservation are typical coordination actions, which can be viewed as indispensible interactions for the corresponding production action (such as Provide Hotel Room) to be triggered. Production action creates a value for Room Guest. It is often the case in practice that the promise or acceptance actions are missing. They can be performed tacitly. For example, there is no acceptance of Pay action in the second interaction loop (see figure 3).

The behavioral effects of communication actions can be expressed by using transition links (—►) between various classes of objects. Reclassification is defined in terms of communication action that is terminating an object in one class and creating it in another class. Sometimes, objects may pass several classes, and then they are terminated. Graphical notation of the reclassification construct is graphically represented in figure 5(a).

Unbroken arrows show control flow of creation and termination effects. Object classes represent a persistent or transient set of objects. Fundamentally two kinds of changes are possible during any reclassification: termination and creation of an object. A creation is denoted by an outgoing transition arrow to a post-condition class. Graphical notation of the creation construct is represented in Figure 5(b). A termination action is represented by the transition dependency directed from a pre-condition object class. Before an object can be terminated, it must be created. A pre-condition

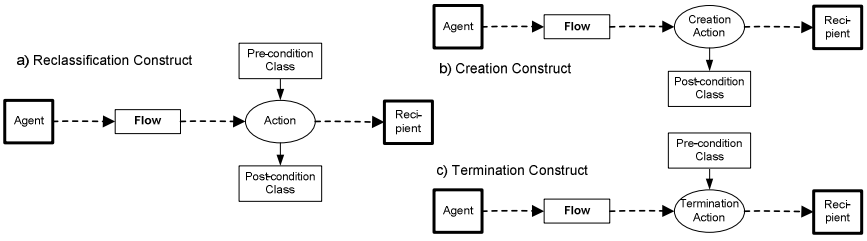


Fig. 5. Graphical representation of three types of modeling constructs

class in the termination construct is understood as final. The graphical notation of a termination action is represented in Figure 5(c).

Behavioral and structural aspects of interactions can be analyzed in terms of their reclassification, creation or termination effects. When two subsystems interact one may affect the state of each other [18]. Structural changes of objects can be defined in terms of object properties [19]. Interaction dependency R(A▪▪▪▪►B) between two active concepts A and B indicates that A subsystem can perform action R on one or more B subsystems. An action typically manipulates properties of one or more objects. Otherwise, this action is not purposeful. Property changes may trigger object transitions from one class to another.

Structural changes of objects are manifested via static and dynamic properties. Dynamic properties are represented as actions, which are connected to classes by creation and termination links. Static properties can be represented by the mandatory attributes. Properties are linked to classes by the single-valued or by multi-valued attribute dependencies. One significant difference of the presented modeling approach is that the association ends of static relations are nameless. Motivation of such way of modeling can be found in another paper [16] by the same author. Semantics of static dependencies are defined by cardinalities, which represent a minimum and maximum number of objects in one class (B) that can be associated to objects in another class (A). Single-valued dependency is defined by the following cardinalities: (0,1;1,1), (0,*;1,1) and (1,1;1,1). Multi-valued dependency denotes either (0,1;1,*) or (1,1;1,*) cardinality. Graphical notation of various static dependencies is represented in figure 6.

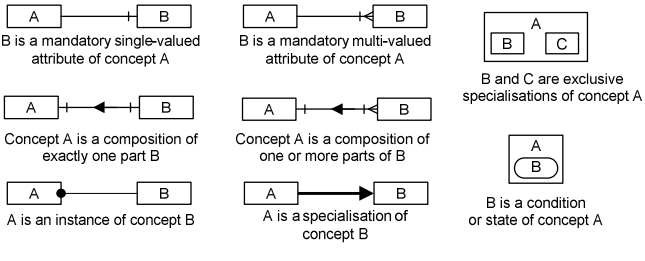


Fig. 6. Notation of static dependencies between concepts

The static dependencies are missing in figure 3. This diagram does not provide any semantic details of control flows between communication actions. It shows only the necessary business events of scenario for staying at a hotel. Both interaction events and transitional effects are necessary to describe the behavior fully. The actions such as Request Room, Create Reservation and Pay should also specify the acceptable ways for structural changes to occur in different classes of objects. The network of loops thus can be interpreted as a set of rights, responsibilities, commitments and claims among various types of actors. If some network segments are unclear or missing, then they may cause breakdowns in business scenarios. Therefore, interaction dependencies can be used to analyze interaction possibilities between actors involved. In general, communication actions can be mandatory, optional, sequential, and alternative or synchronized with the secondary workflow loops.

Triggering conditions of the secondary interaction loops may depend on objects, which are created or terminated in the primary or overlaying workflow loops. Pre-condition and post-condition classes are crucial to understand the dynamic aspects of interactions. Creation or termination of objects allows constructing scenarios, which include optional or mandatory workflows. Static and dynamic dependencies between classes are used to link interaction loops together in different ways. For instance, Provide Hotel Room action consumes Hotel Room[Reserved] object, which is part of Hotel Reservation. The corresponding Reservation object must created in the previous interaction loop (see Create Reservation action). Provide Hotel Room action also creates a Hotel Room[Assigned] object with the property of Room Guest. A preliminary Bill object must be generated for every Room Guest. The Bill object is consumed in the Pay action, which is necessary for creation of Payment. The described creation and termination effects are graphically represented in figure 7.

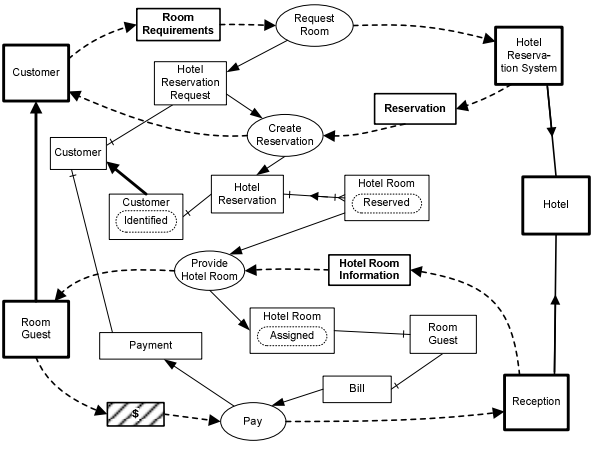


Fig. 7. Examples of creation and reclassification

The presented diagram is superimposing interaction dependencies and behavioral changes in various classes of objects. For instance, the Request Room (R:Request) action represents semantics of creation effects, which are defined by the transition **1—►2** in the conversation for action schema (see figure 4). Create Reservation is a promise (P:Promise). It defines reclassification effects corresponding to the transition **2—►3**. The Provide Hotel Room action represents reclassification effects, which are predefined by the transition **3—►4**. In this way, creation and termination effects define constraints on various types of objects in sending and receiving various types of flows between actors. The diagram visualizes the ways in which different interaction loops are composed. Inheritance, composition and mandatory attribute dependencies can be used for reasoning about the consequences of object creation and termination effects. According to the conceptual modeling rules [16], the termination of Bill object is causing termination of Room Guest and his Hotel Room[Assigned]. It should be noted that the semantic power of UML object flow diagrams is not sufficient for capturing these effects.

A simple interaction loop between service requester and provider can be viewed as the basic element of any communication process [12]. In carrying out the work, a service provider may in turn initiate further interactions. In this way, a network of the loosely coupled actors with various roles comes into interplay to fulfill the original service request. To put it in other terms, the interacting loops can be composed together or overlaid on each other into more complex interaction webs by using creation and termination links. If the object transition effects cannot be conceptualized by the pre-condition or post-condition classes and their properties, then the communication action is not purposeful. Interaction dependencies without purposeful actions make no sense and should be eliminated.

4 Composing Interaction Loops into Scenarios

Interaction loops are specified on different levels of abstraction. They can be combined in various ways into more complex scenarios. The scope of scenario can vary. It may include all business events, or it may include just some events, which are of interest to one specific actor. Scenarios can be used to define workflows on different granularity levels. Our studies indicate that a simple interaction loop can be viewed as fundamental element for composition of scenarios. Every interaction loop can be analyzed separately as it is required by the principle of separation of concerns. In such way, interaction loops provide a natural way of decomposition of processes and data. Two dependent interaction loops can be used for analysis of semantic integrity between static and dynamic aspects. In this way interaction dependencies are crucial to understand a concern composition mechanism. Such graphical descriptions are an excellent means for analyzing the order of interactions.

We will demonstrate how the Reserve Room use-case scenario can be conceptualized by composing three different workflows. The primary interaction loop, which characterizes the Reserve Room use-case functionality (between Customer and Room Reservation System) is represented in figure 7. We will demonstrate conceptualization

of slightly modified scenario, which was analyzed by Jacobson and Ng [5]. It is suitable for the illustration of interaction loop overlays in graphical terms. We assume that the Reserve Room use-case is decomposed into three other use cases, which are represented in figure 8.

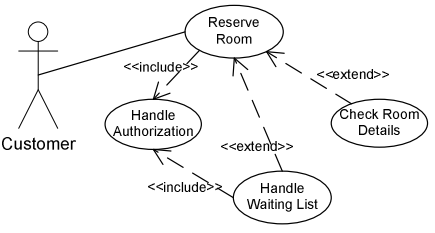


Fig. 8. Example of use case diagram

A Reserve Room use case main workflow scenario can be defined as a sequence of events, which occur during one particular execution of process. This scenario can be described as follows: 1) The Customer **requests room** by entering the specific room requirements (including the desired period of stay), 2) The Hotel Reservation System **offers** various available rooms with different price alternatives, 3) The Customer may Check Room Details, 4) The Customer **selects** the available room (more than one room can be selected), 5) The Hotel Reservation System **creates reservation** with the details of all selected hotel rooms and displays reservation information to the Customer.

We assume that (step 3) Check Room Details should be represented as an optional use case with the extension point in Reserve Room use case. It is defined by the following request and response: 3.1) The Customer **requests room details**, 3.2) The Hotel Reservation System **presents room** information. Reserve Room use-case scenario can be graphically defined by a number of interaction loops between Customer and Hotel Reservation System. The most generic interaction loop can be viewed as overlays of two more specific interaction loops. This scenario describes functionality of two use cases. The first and second interaction loops define the Reserve Room use-case scenario. The third underlying interaction loop represents the functionality of the Check Room Details use-case.

The second interaction loop is synchronized with the Create Reservation action from the primary workflow loop. The overlaying workflow loop is reused from the diagram in figure 7. The secondary underlying interaction loop is describes Customer's service response to Hotel Reservation System's request. It is specified as follows:

If Offer Rooms (Hotel Reservation System ▪▪▪▪▶ Customer)
then Select Room(Customer ▪▪▪▪▶ Hotel Reservation System).

The lowest interaction loop in our example is subsumed by the loop, which is represented in the middle. Both loops are necessary for the selection of desirable room type and for providing necessary data about room guest. In this example, object creation, termination and reclassification effects represent very important semantic details of

unambiguous scenario in which three interaction loops are composed together. The overlaying and the secondary interaction loops are mandatory. The bottom interaction loop is optional. At least the first and second interaction loop should be initiated to the final interaction loop is performed (see the lower loop in figure 7). According to the presented description, Create Reservation is reclassification action, which is composed of Select Room and Offer Rooms actions on the lower level. Moreover, the Select action cannot be triggered prior to Offer Rooms action. Select action can only be performed in parallel with the Create Reservation action, because creation of Hotel Reservation must be created concurrently with the compositional part Hotel Room [Reserved].

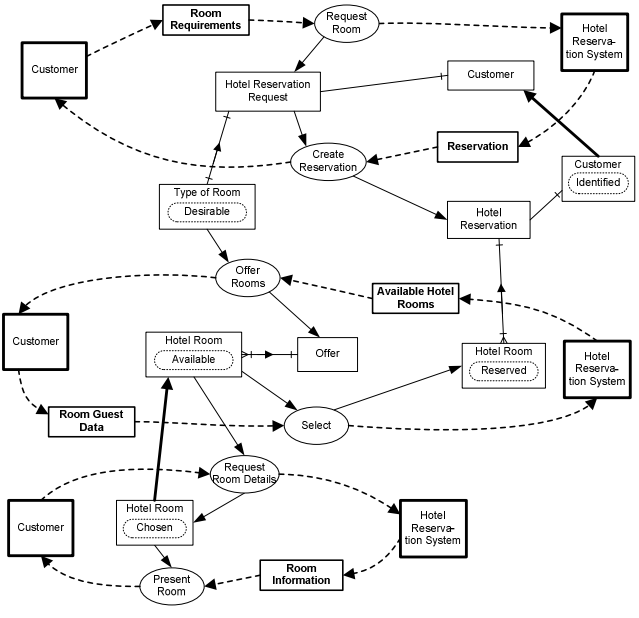


Fig. 9. Graphical representation of scenario by using overlaying interaction loops

The extension use case must have a reference to some extension point indicating when the extension use-case flow will be inserted. There are two alternatives, which are marked by two outgoing unbroken arrows from Hotel Room[Available]. The customer is able to trigger the Select action. It consumes a Hotel Room[Available] object and creates Hotel Room[Reserved] object. Another alternative for Customer is triggering the Request Room Details action. It creates a Hotel Room[Chosen] object. This object is terminated in the Present Room action. A customer may Request Room Details any number of times for each available room and then to Select it. So, the most specific underlying loop is optional and it can be executed multiple times.

The Handle Waiting List use-case (see figure 8) requires definition of an extension point in the Reserve Room use-case as well. The alternative scenario is represented in figure 10.

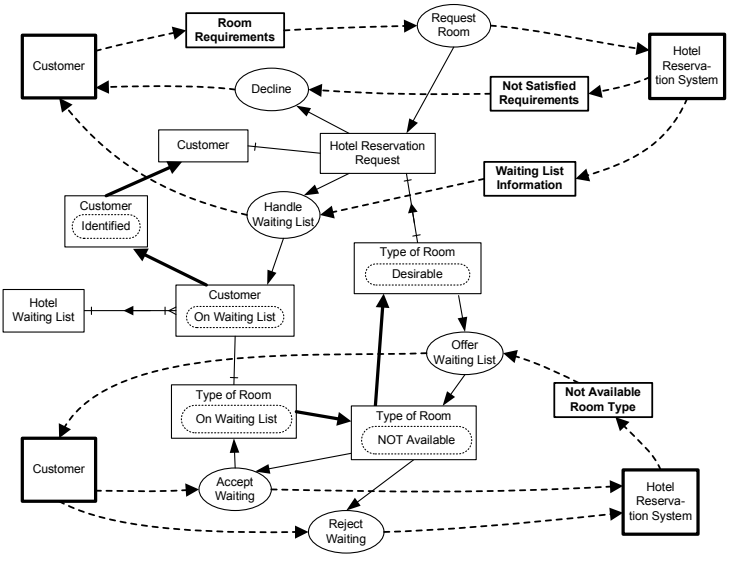


Fig. 10. Example of an alternative workflow

The Offer Rooms action in the main scenario can be performed successfully if and only if one or more desirable rooms are available. Possibility of failure to create an Offer with at least one Hotel Room[Available] (see figure 9), requires definition of the alternative flow, which is represented by the Offer Waiting List action. Handle Waiting List action is refined in terms of two more specific actions such as Offer Waiting List and Accept Waiting. These two actions are included in the underlying interaction loop. Handle Waiting List use-case extends the Reserve Room use-case when an Offer object cannot be created, because a desirable hotel room is not available. If Type of Room[NOT Available] object is created, it preserves from termination (see inheritance) the desirable type of room in the Hotel Reservation Request. Yet another alternative is to terminate the Hotel Reservation Request by Decline action. This option may be caused by a failure of the Handle Waiting List action. Please note that Customer[On Waiting List] object can be created just in case if a Customer agrees to Accept Waiting. If Customer triggers the Reject Waiting action, then the Handle Waiting List action will fail.

Dependent classes can be used as joint points for inclusion of the mandatory flows, which must be executed across different scenarios. One example of such flow is represented by the Handle Authorization use-case (see figure 8). Authorization flow is necessary for creation of Customer[Identified] object (see figure 10) in Handle Waiting List use-case. Creation of this object must take place in the Reserve Room use-case as well. The semantics of the authorization flow is defined by one underlying interaction loop in figure 11.

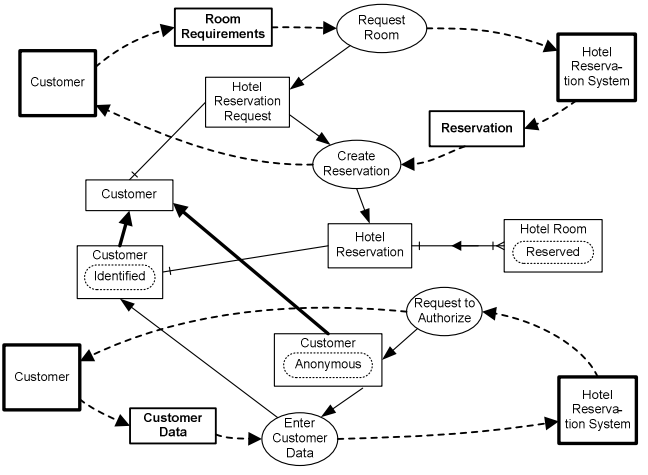


Fig. 11. Composition of two interaction loops to Handle Authorization and Reserve Room

Customer[Identified] class represents a subset of logged-in customers. Creation of a logged-in customer is mandatory in two other overlaying interaction loops (see figure 9, 10). Customer[Identified] can be understood as a joint point of two different scenarios. In our example, two overlaying interaction loops require the authorization to be completed (see Reserve Room and Handle Waiting List use-cases).

5 Concluding Remarks

The traditional conceptual modeling methods are used for the analysis of business processes and business data in isolation. In this situation, it is difficult to achieve semantic integrity of static and dynamic aspects when several crosscutting concerns are combined together. Inability to detect integrity problems of various concerns in early system development stages is one source of errors in IS specifications. Since the realization of use cases touches several classes, use-case-driven approach is a dominant concern separation technique. However, it is very difficult to integrate crosscutting concerns by using UML diagrams, because the static and dynamic aspects of conceptual models are not explicitly linked to use-cases. Most graphical modeling techniques are not flexible for the analysis of interplay among behavioral, interactive and structural aspects. We have demonstrated by examples how sequential, iterative, parallel and alternative behavior can be captured by using interaction dependencies between actors. Isolated interaction loops can be composed together by using creation, termination and reclassification constructs.

Separation of crosscutting concerns in terms of simple interaction loops is important for designers to construct scenarios with a more understandable structure. If system architects are not able to separate concerns, the complexity of analysis and design task increases exponentially and it is difficult to meet evolving needs of stakeholders. Introducing underlying sub-flows and corresponding alternate flows in

scenario specifications help designers to achieve much desired modularity. Comprehensible structure is important for diagnosing potential problems in business scenarios. For instance, every simple interaction loop can be used for analyzing breakdowns in the main business scenario. Unexpected breakdowns contribute to working overload and give rise to customer complaints. A breakdown potentially creates a situation when a service requester is either expecting work that provider will not do or in contrary - doing some service that a requester is not interested to receive. Missing alternative interaction loops are semantic holes of conceptual representations. Such incompleteness can be viewed as hidden requirement engineering defects, which may potentially inject mistakes or ambiguities of conceptualizations.

Separation of crosscutting concerns and composition of interaction loops together suggests a flexible way for managing complexity of conceptual representations. We have demonstrated by examples some basic principles of the non-traditional conceptual modeling approach, which allows designers to visualize and to analyze semantic integrity of conceptual representations of scenarios. By using the presented set of semantic dependencies, the underlying interaction loops can be enhanced or redesigned on demand. The integrated graphical modeling method is useful for analysis of interplay among interactive, behavioral and structural aspects of conceptualizations. Desired modularity is achieved by decomposing complex scenarios into simple underlying or overlaying interaction loops. Semantically integrated conceptual descriptions of scenarios are easier to understand, extend and maintain.

References

1. Gane, C., Sarson, T.: Structured System Analysis. Prentice-Hall, NJ (1979)
2. Yourdon, E., Constantine, L.L.: Structured Design. Prentice-Hall, NJ (1979)
3. OMG. Unified Modeling Language Superstructure, version 2.2 (2010) , http://www.omg.org/spec/UML/2.2/ (retrieved January 19, 2010)
4. Snoeck, M., Dedene, G., Verhelst, M., Depuydt, A.M.: Object-Oriented Enterprise Modelling with MERODE. Leuven University Press (1999)
5. Jacobson, I., Ng, P.-W.: Aspect-Oriented Software Development with Use Cases. Pearson, Pennsylvania (2005)
6. Gordijn, J., Akkermans, H., van Vliet, H.: Business modelling is not process modelling. In: Mayr, H.C., Liddle, S.W., Thalheim, B. (eds.) ER Workshops 2000. LNCS, vol. 1921, pp. 40–51. Springer, Heidelberg (2000)
7. Wagner, G.: The Agent-Object-Relationship Metamodel: Towards Unified View of State and Behaviour. Information Systems 28, 5 (2003)
8. Dietz, J.L.G.: DEMO: Towards a Discipline of Organisation Engineering. European Journal of Operational Research, 351–363 (2001)
9. Zachman, J.A.: A Framework for Information System Architecture. IBM Systems Journal 26(3) (1987)
10. Glinz, M.: Problems and Deficiencies of UML as a Requirements Specification Language. In: Proc. of the 10-th International Workshop on Software Specification and Design, San Diego, pp. 11–22 (2000)
11. Dori, D.: Object-Process Methodology: A Holistic System Paradigm. Springer, Berlin (2002)
12. Denning, P.J., Medina-Mora, R.: Completing the Loops. Interfaces 25, 42–57 (1995)

13. Gustas, R.: Conceptual Modeling and Integration of Static and Dynamic Aspects of Service Architectures. In: International Workshop on Ontology, Conceptualization and Epistemology for Information Systems, Software Engineering and Service Sciences, Hammamet, Tunisia, pp. 17–32. Springer, Heidelberg (2010)
14. Gustas, R., Gustiene, P.: Pragmatic – Driven Approach for Service-Oriented Analysis and Design. In: Information Systems Engineering - from Data Analysis to Process Networks. IGI Global, USA (2008)
15. Ferrario, R., Guarino, N.: Towards an Ontological Foundation for Services Science. In: Domingue, J., Fensel, D., Traverso, P. (eds.) FIS 2008. LNCS, vol. 5468, pp. 152–169. Springer, Heidelberg (2009)
16. Gustas, R.: Modeling Approach for Integration and Evolution of Information System Conceptualizations. International Journal of Information System Modeling and Design 2(1), 45–73 (2011)
17. Winograd, T., Flores, R.: Understanding Computers and Cognition: A New Foundation for Design, Ablex Norwood, N.J. (1986)
18. Evermann, J., Wand, Y.: Ontology Based Object-Oriented Domain Modeling: Representing Behavior. Journal of Database Management 20(1), 48–77 (2009)
19. Bunge, M.A.: Treatise on Basic Philosophy. Ontology II: A World of Systems, vol. 4. Reidel Publishing Company, Dordrecht (1979)

Method Families Concept: Application to Decision-Making Methods

Elena Kornyshova, Rébecca Deneckère, and Colette Rolland

Centre de Recherche en Informatique, Université Paris 1 Panthéon-Sorbonne,
90 rue de Tolbiac, 75013 Paris, France
{kornyshova,denecker,rolland}@univ-paris1.fr

Abstract. The role of variability in Software engineering grows increasingly as it allows developing solutions that can be easily adapted to a specific context and reusing existing knowledge. In order to deal with variability in the method engineering (ME) domain, we suggest applying the notion of method families. Method components are organized as a method family, which is configured in the given situation into a method line. In this paper, we motivate the concept of method families by comparing the existing approaches of ME. We detail then the concept of method families and illustrate it with a family of decision-making (DM) methods that we call MADISE.

Keywords: Situational Method Engineering, Method Family, Method Line, Decision-Making Methods.

1 Introduction

An information system development methodology is a set of ideas, approaches, techniques and tools used to transform organizational needs into an appropriate information system (IS). There are many and various application domains for these methodologies. However, because of this diversity, it is now clear that a universal method that could be applied to handle completely any IS development project does not exist. Method Engineering (ME) is a discipline which aims to bring effective solutions to the construction, improvement and modification of methods. Several authors tried to conceive methods that would be as effective and as adapted as possible to the IS development [1] [2]. However, this objective was not always reached, especially because the methods were not really adapted to project situations. The situational methods were designed to correct this drawback. Situational Method Engineering (SME) finds its justification in the practical field analysis which shows that a method is never followed literally [3] [4]. It promotes the idea of using combined method parts, instead of complete methodologies, to specific situations [5].

However, these approaches are not widely spread in the practitioners' world. The components composition is a quite complicated process as there may exist some overlapping between concepts (which is the rationale behind the integration process of the assembly technique) and no SME approach really offers a simple and easily understandable way to construct a situational method. Furthermore, it is nowadays

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 413–427, 2011.

acknowledged that contingency factors change continuously during the project life cycle imposing a continuous change management. This last feature raises the problem of the dynamic adaptation of methods, which has not been considered by current SME approaches [6].

Our proposition suggests to move away from this construction of methods 'on the fly' to the management of a set of similar components as a whole. Our proposal is to organize these components in method families to manage variability and commonalities in order to promote the reuse and the adaptability of method families. The method family is then configured in a given project in order to obtain an appropriate method line. We think that method families do exist today in companies and could beneficially be handled in an easier manner.

This paper is organized as follows. In Section 2, we present a brief state-of-the-art of the seven most known SME approaches in order to identify their drawbacks. We offer a general vision of the method family concept and describe its model in Section 3. We illustrate our proposal with the example of MADISE, a family of decision-making methods in Section 4. We conclude in Section 5.

2 SME Approaches: State-of-the-Art

In this section, we present the comparison of the seven main existing SME approaches and we give an analysis of their drawbacks.

The four views framework has proved its efficiency in enhancing the understanding of various engineering disciplines such as information systems engineering, requirements engineering, IS development process engineering and method engineering. Our point of view is that this framework concept can be used to help in understanding and comparing different SME approaches.

For our purpose, we define the SME four-dimensional framework as follows:

- **Objective view.** In the objective view, the corresponding dimension allows investigating the rationale of SME approaches.
- **Subject view.** This view expresses the dimension which deals with the representation of SME approaches, their nature.
- **Development view.** The development view deals with the process of constructing the SME approaches.
- **Usage view.** This view deals with different aspects that describe the SME approaches usage.

We propose a review of seven SME approaches. We choose our method panel in the set of the most widespread approaches and with the intention to offer a more complete study of the different views: Method fragment approach [7] [8] [9], Method Chunk Approach [3] [11], Method Configuration Approach (Component) [12] [13] [14], OPEN Process Framework [15] [16], Method Service Approach [17], Method Extension Approach [18], and FIPA (Foundation for Intelligent Physical Agent) approach [19] [20] [21] [22]. These seven approaches are compared according to the suggested framework. Table 1 resumes the results of this comparison (The detailed description of the SME approaches comparison can be found in [23]).

Table 1. SME Approaches' Review according to the Framework

View	Attribute	Value	Method fragment	Method chunk approach	Method configuration approach	OPF approach	SO2M, method service approach	Method extension approach	FIPA approach
Objective	Covered way	{Way of thinking, Way of modeling, Way of organising, Way of supporting}	{Way of thinking, Way of modeling, Way of organising}	{Way of thinking, Way of modeling, Way of organising}	{Way of thinking, Way of modeling, Way of organising}	{Way of thinking, Way of modeling, Way of organising}	{Way of thinking, Way of modeling, Way of organising}	{Way of thinking, Way of modeling, Way of organising}	{Way of thinking, Way of modeling, Way of organising}
	Target issues	{Variability, Intentionality, Context-Awareness, Reusability, Conflict Resolution}	{Context-Awareness, Reusability}	{Intentionality, Context-Awareness, Reusability}	{Intentionality, Context-Awareness, Reusability}	{Context-Awareness, Reusability}	{Intentionality, Context-Awareness, Reusability}	{Intentionality, Context-Awareness, Reusability}	{Intentionality, Context-Awareness, Reusability}
Subject	Actor representation	{Role, Producer, User}			{Role}	{Role, Producer}			{Role}
	Knowledge dependency	{Yes, No}	{Yes}	{Yes}	{Yes}	{Yes}	{Yes}	{Yes}	{Yes}
	Knowledge representation	{Fragment, Chunk, Component, OPF Fragment, Service, Pattern}	{Fragment}	{Chunck}	{Component}	{OPF fragment}	{Service}	{Pattern}	{Fragment}
	Variability representation	{Yes, No}	{No}	{No}	{No}	{No}	{No}	{No}	{No}
	Context representation	{Reuse frame, Interface, Contingency factor, Development situation}	{Contingency factor}	{Reuse frame, Interface}	{Development situation}		{Interface}	{Interface}	{Development situation}
	Abstraction Level	{Conceptual, Technical}	{Conceptual}	{Conceptual}	{Conceptual}	{Technical}	{Technical}	{Conceptual}	{Conceptual}
Development	Knowledge construction	{Ad-hoc, Formalised}		{Formalised}	{Formalised}	{Formalised}		{Formalised}	{Ad-hoc}
	Reengeneering process	{Decomposition, Assembly, Instantiation}	{Decomposition}	{Decomposition}	{Decomposition}	{Decomposition}	{Decomposition}	{Instantiation}	{Decomposition}
	Knowledge organisation	{Repository, Organisational process}	{Repository}	{Repository}	{Repository, Organisational process}	{Repository}	{Repository, Organisational process}	{Repository, Organisational process}	{Repository}
	Context specification	{Yes, No}	{No}	{No}	{No}	{No}	{No}	{No}	{No}
	Tool/ Implementation	{Product storage and manipulation, Process operating, Retrieval, Construction}	{Product storage and manipulation, Retrieval, Construction}	{Product storage and manipulation, Retrieval}	N/S	N/S	{Product storage and manipulation, Retrieval, Construction}		
Usage	Process model	{Activity-oriented, Product-oriented, Decision-oriented}	{Activity-oriented}	{Decision-oriented}	{Activity-oriented}	{Activity-oriented}	{Decision-oriented}	{Decision-oriented}	{Activity-oriented}
	Construction flexibility	{Selection and adaptation of a method, Modular construction of a method, Method selection in a multi-method}	{Modular construction of a method}	{Modular construction of a method}	{Selection and adaptation of a method, Modular construction of a method}	{Modular construction of a method}	{Modular construction of a method}	{Selection and adaptation of a method}	{Modular construction of a method}
	Construction technique	{Assembly, Instantiation, Extension, Reduction, Agile Construction}	{Assembly}	{Assembly, Extension}	{Assembly, Extension, reduction}	{Assembly, Instantiation, Agile construction}	{Assembly}	{Instantiation, Extension}	{Assembly}
	Context Usage	{Project characterisation, Component characterisation, matching component with situation}			{Project characterisation}	{matching component with situation}	{Project characterisation, Component characterisation, matching component with situation}	{Component characterisation, matching component with situation}	{Component characterisation}

The framework analysis allows identifying the following main drawbacks of the studied SME approaches:

- **Approaches interoperability.** Despite some standardisation efforts of the ME community, all approaches are strongly coupled with their own notion of method component so the techniques developed in one approach are not usable in another one.
- **Component retrieval.** As there is no common interface between components, their retrieval is made dependent of their nature. Locating the needed component may require searching several repositories rather than having them in one centrally available place [6]. Moreover, most of the approaches don't propose an organisational process to handle the components, which complicate their retrieval in the execution of the construction process.
- **Variability.** The variability issue is not taken into account in any approach so there is no representation of the common or variables parts in the current approaches.
- **Contextualization.** In order to use the context variability in an optimistic way, it is necessary to have the three parts of the context usage: the project characterisation (to describe the development situation, which evolves continuously), the component characterisation (to describe the reuse context) and the matching between them (to be able to choose the appropriate component in the appropriate situation). Only the method service approach proposes this contextualisation. In addition, none of the existing SME approaches does suggest a methodology for defining the context of methods.

3 Method Family Concept

This section offers the method family model, the general vision of the method family construction and usage, and the method family organization.

3.1 Method Family Model

The concept of method family is described with the meta-model of Figure 1 (using the UML formalism).

The ***method component*** concept has the same semantic that the classic SME methods (a building block, subset of a method), which may contains other components.

We propose in this work the concept of ***method family***, which is a set of several organized method components for a specific domain.

In a method family, some components may be considered either as ***common*** or ***variable*** as each situation may require specific components. These situations, where it is necessary to choose between several components are called ***variation point***. The relationship between the variation point and the component defines the ***variability dependency,*** which can be ***mandatory*** or ***optional*** (and respectively corresponds to the common and variable method components). A set of variable method components represent the alternatives offered to the engineer at a specific variation point, it is a way to realize variability. This representation of variability was inspired from [24] which discusses the variability of software product lines.

The ***method component context*** must be specified. It characterizes all possible situations in which this component may be applied. The ***project context*** includes all characteristics of the situation at hand, which matches some variable method component contexts. These two context concepts are specified on the same basis as they inherit from the ***context***. This allows a better matching between the situation and the components for the configuration.

A variable method component is selected following the situation at hand (i.e. the project context). The set of these ***selected method components***, together with the common method components, represents a ***project method line***.

A ***method line*** is either an ***initial method*** (a method already known) or a ***project method line*** (defined with our process). The project method line includes the mandatory components and those selected for the given project based on the project characteristics. A method family is composed of a set of method lines. This concept allows regrouping several method lines for a specific domain. Each method line or method family may itself be considered as a method component.

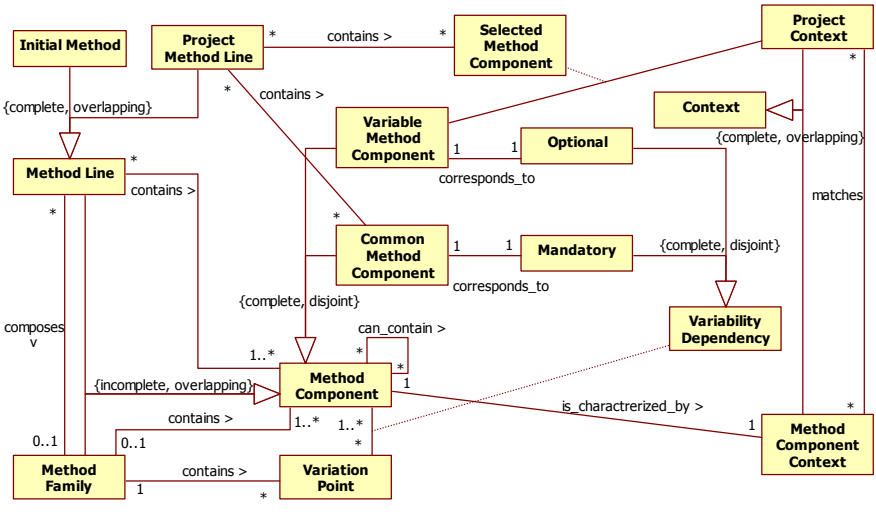


Fig. 1. Method Family Meta-Model

Based on the comparison framework, the method family is characterized as follows:

```
Objective View
   Covered way = {Way of thinking, Way of modeling, Way of
      organizing}
   Target issues = {Variability, Intentionality, Context-Awareness,
      Reusability, Conflict Resolution}
Subject View
   Actor representation = {Role, User}
   Knowledge dependency = {No}
   Knowledge representation = {Fragment, Chunk, Component, OPF
      Fragment, Service, Pattern}
   Variability representation = {Commonalities and variability}
```

```
    Context representation = {Method service context, Contingency
       factor, Development situation}
    Abstraction Level = {Conceptual}
Development View
    Knowledge construction = {Formalized}
    Reengineering process = {Decomposition, Assembly}
    Knowledge organization = {Repository, Organizational process}
    Context specification = {yes}
    Tool/ Implementation = {Product storage and manipulation, Process
       operating, Retrieval, Construction}
Usage View
    Process model = {Decision-oriented}
    Construction flexibility = {Method selection in a multi-method}
    Construction technique = {Agile construction}
    Context Usage = {Project characterization, Component
       characterization, matching component with situation}
```

Thus, we can consider that the concept of method family allows overcoming the established drawbacks:

- This approach is independent of the knowledge representation as the concept of family may be applied to any component type (fragment, chunk, component and so on).
- A method family organizes components of the same domain in a way which enables their easier usage as it is already a subset of the potential method components. Then, the method family is configured according to a project needs. It allows avoiding the retrieval step (in several method bases) and composition step (as the method family is already composed of the method components).
- A method family is based on the separation of common and variable components in each variation point. This facilitates the use of SME approaches in practice.
- The given approach covers all necessary elements for the contextualization as it handles project characterization, component characterization, and matching component with situation.

3.2 General Vision of Method Family Construction and Usage

A method family is constructed based on the existing methodologies in a given field and for their further usage in different projects. Figure 2. gives an overview of method families and their usage.

The first step is to construct the method family from the existing methods. The next step is to configure the method family in order to obtain a method line adapted to the specific conditions of the project at hand. Finally, the obtained method line is applied in this project. On this basis, we define three following processes:

- Method family definition process;
- Method line configuration process;
- Method line application process.

Method family definition process. This process allows constructing method family from existing methods. Existing methods are decomposed into modular components (like in different SME approaches) and are combined into the same model (i.e. a *family*). The combination of method components follows two main principles:

- Identification of variation points, common and variable components;
- Identification of components having the same goal and the same target product, but different ways of working.

At this step, the variability is taken into account as common and variable parts are shown. Alternative method components are identified and organized as variants in a given variation point. The difference with the usual SME approaches is that we combine all the components of the same domain in order to obtain a more 'generic' method (the family).

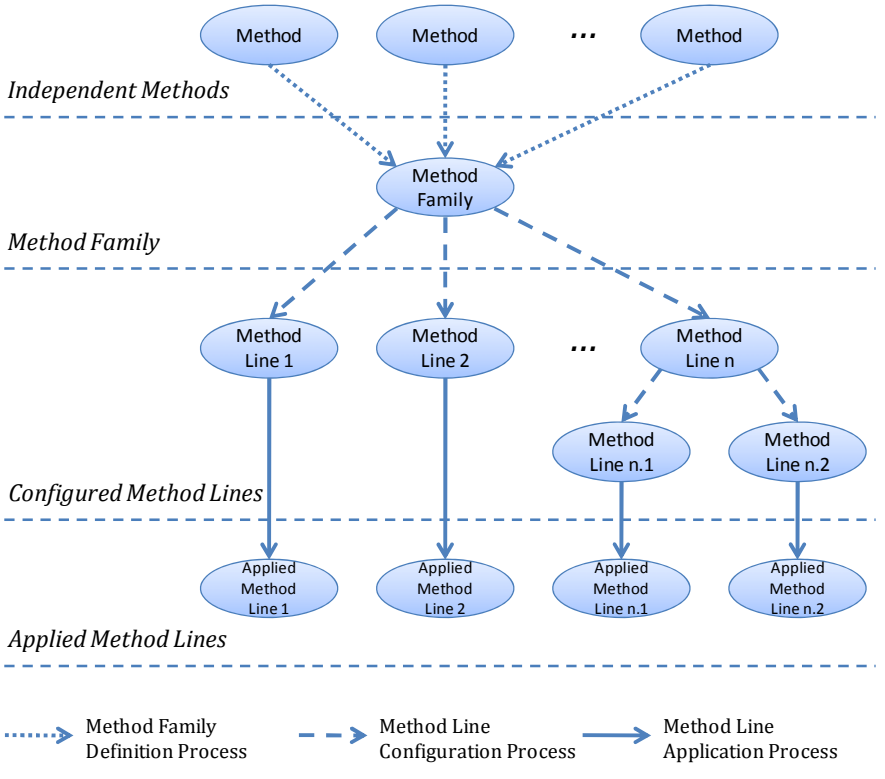


Fig. 2. General Vision of Method Families

Method line configuration process. A method line is obtained following the configuration of the family according to various criteria.

We suggest three kinds of configuration:

- ***Complete method line selection.*** This configuration type helps to select between all the possible method lines.
- ***Method components sub-set selection.*** This kind of configuration allows to select a sub-set of components based on the context characteristics in a given project.

- ***Step by step method components selection.*** In this configuration type, the method line is configured during the project realization as the components are selected one by one.

Variation points facilitate the configuration process as common and variable components and the definition of their context enable to configure the method family according to the project needs. These kinds of configurations were described in [25]. It is also possible to run multiple sequential configurations to acquire the desired method line.

Method line application process. Once the method line has been created, a final configuration can be applied to obtain the method that will be used in a specific case.

The method family usage aims at constructing a method 'on the fly', following the project characteristics. However, as usual SME approaches deal with the construction process itself and its difficulties (integration *versus* association, for instance), method families offers a way to simplify the work of practitioners with a decomposition of the construction process. The method engineer constructs the method family but the practitioner just has to configure the method family to obtain a specific method adapted to his needs. The construction of the large method family base (repository) is justified by the need to provide a more flexible and context-aware usage of methods belonging to the same domain.

3.3 Method Family Organization

We use the MAP formalism [26] for representing method family and for organizing method components within method families.

A map is presented as a graph where nodes are *intentions* and edges are *strategies*. The key concept of a Map is the notion of the section which is an aggregation of two specific intentions, the *source intention* and the *target intention*, linked together with a *strategy*. It embeds the knowledge corresponding to a particular method component to achieve an intention (the target intention) from a specific situation (the source intention) following a particular technique (the strategy). When dealing with method families modeled by maps, each method component is represented by a map section, as shown in Figure 3.

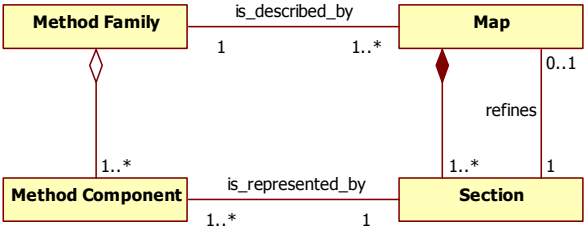


Fig. 3. Map and Method Family Correspondence

This kind of organization provides the mean for ensuring the variability within method families.

4 Method Family Application to Decision-Making Methods

In this section, we show MADISE DM Method Family and two method lines obtained from this family.

4.1 MADISE Decision-Making Method Family

The MADISE DM Method Family describes the generic DM process including the main activities used for DM. As in the previous section, we have selected the Map formalism for representing the family of DM methods. The DM map is a collection of DM method components organized into a family in order to allow its further configuration according to a given situation. The MADISE DM Method Family modelled with MAP is presented in Figure 4.

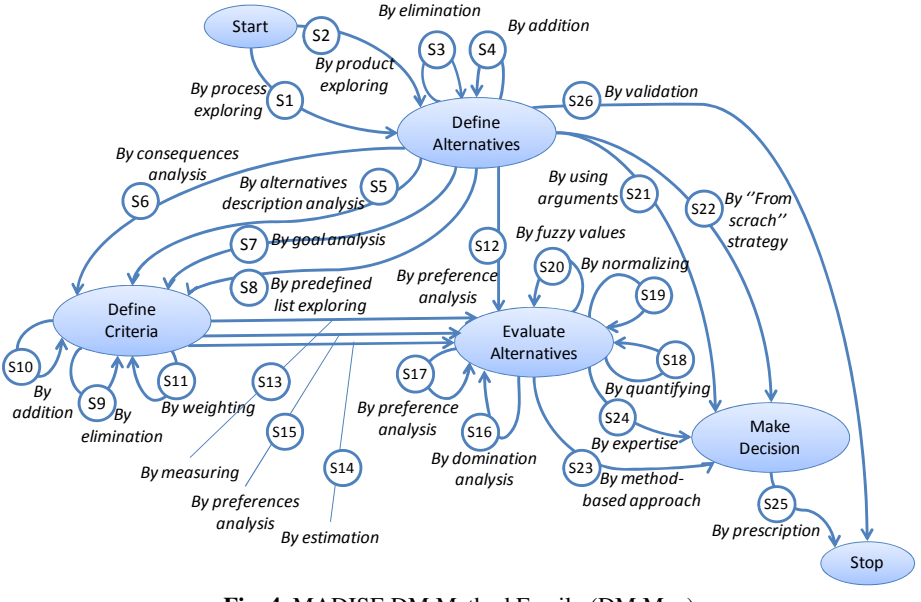


Fig. 4. MADISE DM Method Family (DM Map)

The usage of the DM Family is characterized by data which are required for starting and finishing the corresponding process. This implies the identification of the Input and Output data. Two kinds of information are required before beginning the use of the DM Map: the decision object and the decision problem which are *Input data*. The *Output data* is a decision made according to the identified decision problem. The DM output could also have a NULL value if the decision is not made (for different reasons, such as a lack of information, not valid alternatives etc.). Figure 5. summarizes the Input and Output Data.

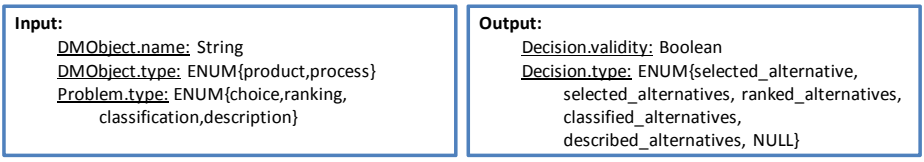


Fig. 5. Input and Output Data of the DM Method Family

The DM Map contains four main intentions: Define Alternatives, Define Criteria, Evaluate Alternatives, and Make Decision.

The engineer starts the MADISE process by reaching the *Define Alternatives* intention. At this stage, an alternative set (or alternative family) is generated.

The *Define Criteria* intention is not mandatory. The engineer selects it if he wants to arbitrate between alternatives based on multiple factors. At this stage, a set of criteria for alternatives evaluation is defined, in particular only one criterion.

The *Evaluate Alternatives* intention aims at constructing the evaluation matrix (or decision matrix) [27].

At the *Make Decision* stage, a prescription for a decision is made.

4.2 Decision-Making Method Lines

In this section, our goal is to show that existing DM processes could be expressed through the MADISE DM method family. For doing this, each DM process must be represented as a MADISE line (i.e. a sub-set of MADISE sections). In order to illustrate this, we have chosen two existing and well-known DM processes: the cost-value approach for requirements prioritization [28], and tool selection from the Rational Unified Process (RUP) [29]. Their DM processes are captured and expressed as method lines in the following sub-sections.

Application Case: the Cost-Value Requirements Prioritization Approach. The cost-value requirements prioritization approach [28] aims at ranking requirements using the AHP DM method. The AHP (Analytic Hierarchy Process) proposed per T.L. Saaty [30]. As a shot reminder, this method is based on pair-wise comparison between alternatives and/or criteria and aggregation of comparison results into a quantitative indicator (score).

Figure 6. shows the DM method line corresponding to the cost-value approach.

The cost-value approach trajectory through the MADISE Map is as follows. The product based strategy is available for identifying candidate requirements (the *By product exploring* strategy is selected). This approach suggests reviewing candidate requirements for ensuring their completeness and correctness. Therefore, requirements can be added to or removed from the initial set (The *By elimination* and *By addition* strategies are selected). The approach defines two criteria describing requirements: relative cost and relative value. These criteria are predefined by the cost-value approach (The *By predefined list exploring* strategy is selected). Actors (users and engineers)

express their preferences by pair-wise comparison for defining the relative value and cost of candidate requirements (The *By preferences analysis* strategy is selected). The aggregated value obtained by AHP application is used for ranking requirement. The cost-value approach uses also a cost-value diagram in order to assist DM (The *By method-based approach* strategy is selected). A consistency index is calculated in order to check the result validity (The *By prescription* strategy is selected). The DM components used by the cost-value approach are resumed in Table 2.

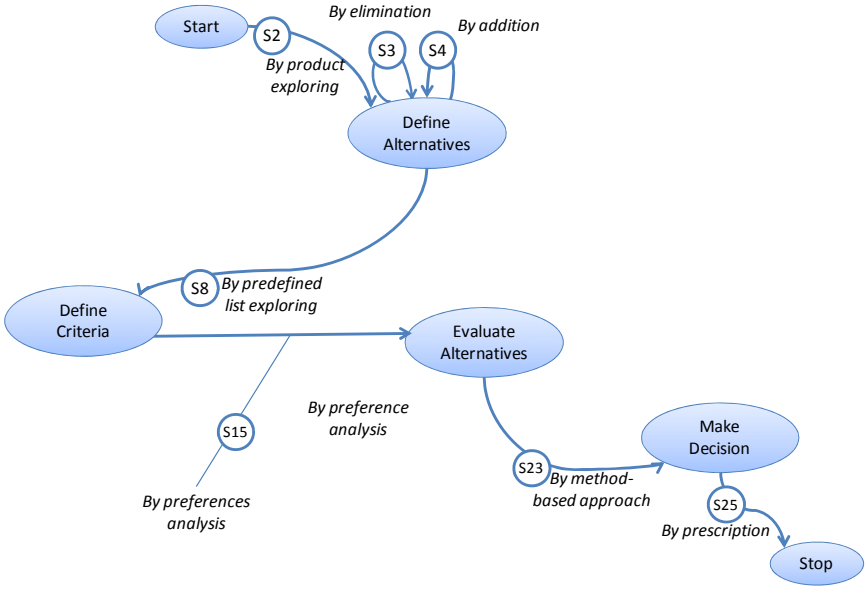


Fig. 6. DM Method Line of the Cost-Value Requirements Prioritization Approach

Table 2. DM Method Line of the Cost-Value Approach: DM Components List

Section	Component Name	Component Signature
S2	Define alternatives list by product exploring	<Start, By product exploring, Define Alternatives>
S3	Refine alternative list by elimination	<Define Alternatives, By elimination, Define Alternatives>
S4	Refine alternative list by addition	<Define Alternatives, By addition, Define Alternatives>
S8	Define criteria by predefined list exploring	<Define Alternatives, By predefined list exploring, Define Criteria>
S15	Evaluate alternatives by preferences analysis according to a criterion	<Define Criteria, By preferences analysis, Evaluate Alternatives>
S23	Make decision by method-based approach	<Evaluate Alternatives, By method-based approach, Make Decision>
S25	Prescribe decision	<Make Decision, By presctiption, Stop>

Figure 7. shows the input and output data for the DM method line corresponding to the cost-value approach. The DM object is *requirement* which is of the *product* type. The problem type is *ranking*. The output is the validated decision representing ranked requirements.

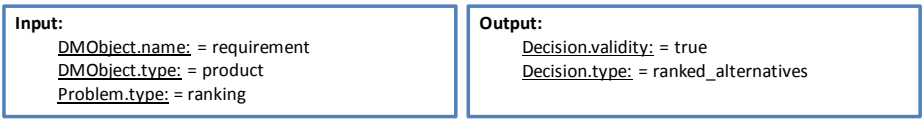


Fig. 7. Input and Output Data of the DM Method Line of the Cost-Value Requirements Prioritization Approach

Application Case: the Tool Selection in RUP. The second example deals with tool selection and was taken from the RUP [29]. The RUP provides a wealth of guidance on software development practices. One of these practices is "Select and Acquire Tools". This task guides the selection of tools that fit project needs.

The DM method line representing the tool selection in RUP is illustrated at Figure 8.

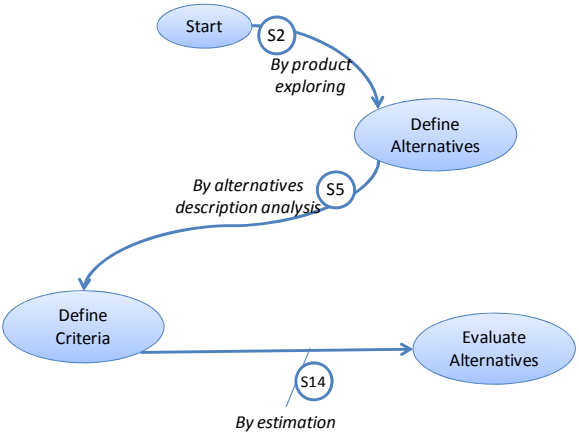


Fig. 8. DM Method Line of the Tool Selection in RUP

The tool selection trajectory includes the following steps. A tool to select is considered as a product (The *By product exploring* strategy is selected). One of the steps in this task is to collect information about tools in order to decide which tool is suitable for the project at hand. The suggested criteria are: (i) tool criteria (features and functions, integration, applicability, extendibility, team support, usability, quality, performance, maturity); (ii) vendor criteria (stability, support availability, training, availability, growth direction); (iii) cost (acquisition cost, implementation cost, maintenance cost). These criteria are based on the tools description (The *By alternatives description analysis* strategy is selected). The RUP proposes to grade each criterion for evaluating candidate tools. The engineer estimates tools according to different

scales. Therefore, the evaluation is subjective (The *By estimation* strategy is selected). The recommendations of the RUP methodology stop at this stage. RUP does not contain any method for aggregating evaluations. Table 3. shows the set of the DM components retrieved in the RUP tool selection task.

Table 3. DM Method Line of the RUP Tool Selection: DM Components List.

Section	Component Name	Component Signature
S2	Define alternatives list by product exploring	<Start, By product exploring, Define Alternative>
S5	Define criteria by alternatives description analysis	<Define Alternative, By alternatives description analysis, Define Criteria>
S14	Evaluate alternatives by estimation	<Define Criteria, By estimation, Evaluate Alternatives>

Figure 9. illustrates the input and output data used in the RUP tool selection DM method line. The DM object is *tool,* which is a *product*. The goal is to select a tool. Therefore, the problem type is *choice*. The output is the validated decision, which corresponds to a selected tool.

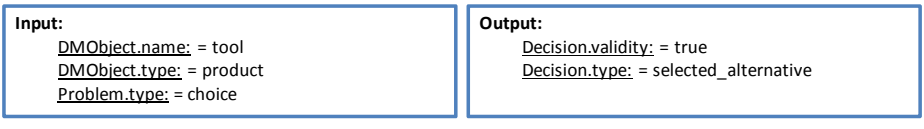


Fig. 9. Input and Output Data of the DM Method Line of the Tool Selection in RUP

As we can see, the cost-value approach provides a more detailed guideline for DM. It allows a complete DM process from the alternatives definition to the decision validation. It contains a possibility to dynamically adjust an alternatives set. Two examples have different strategies for evaluating alternatives. The tool selection approach is simpler to carry out but it does not contain any decision-making and validation steps (we can see that there is no corresponding DM component in the DM method line).

Both examples are expressed as DM method lines as we have identified a suitable trajectory in the DM method family for each of them. The two DM processes are completely covered by the MADISE approach. Therefore, these examples help in validating the MADISE DM method family.

Moreover, another aspect is highlighted within these examples as the DM method family can contribute to improving existing DM models. For instance, the tool selection approach does not provide any advice for aggregating values, as mentioned above. Therefore, it can be completed by the DM component *Make decision by method-based approach* (the <Evaluate Alternatives, By method-based approach, Make Decision> section) in order to include an aggregation method. In the same way, the cost-value approach may be enhanced by the DM component *Discard alternatives by domination analysis* (the <Evaluate Alternatives, By domination analysis, Evaluate

Alternatives> section) in order to eliminate dominated alternatives and, in this way, to simplify the AHP method application.

5 Conclusion

Following a brief state of the art of the main existing SME approaches, we have identified some drawbacks which may explain the reluctance of practitioners to use these SME approaches. We propose in this work a way to solve some of them with the notion of method family. We introduce the method family concept and we illustrate its application with the decision-making methods and method lines.

Method families help to organize a set of components for a specific domain. Thus, the engineer selects a specific method line, inside a family, to apply on its specific project. This approach allows using interoperable components, organized for a specific domain, based on the situation context.

Our future research includes validating this method family approach for all its steps: method family construction for several domains, method line configuration for different specific processes and method line application for several case studies. The MADISE Decision-making family is currently under evaluation by researchers and practitioners from different fields.

References

1. Firesmith, D., Henderson-Sellers, B.: The OPEN Process Framework. An Introduction. Addison-Wesley, Reading (2001)
2. Rolland C., Cauvet C.: Object-Oriented Conceptual Modelling, CISMOD 1992, International Conf. on Management of Data, Bangalore (1992)
3. Ralyte, J.: Method chunks engineering, PhD thesis, University of Paris 1-Sorbonne (2001)
4. Mirbel, I., De Rivieres, V.: Adapting Analysis and Design to Software Context: The jecko Approach. In: 8th International Conference on Object Orirented Information Systems (2002)
5. Ralyté, J., Rolland, C.: An approach for method reengineering. In: Kunii, H.S., Jajodia, S., Sølvberg, A. (eds.) ER 2001. LNCS, vol. 2224, p. 471. Springer, Heidelberg (2001)
6. Rolland, C.: Method engineering: towards methods as services. Software Process: Improvement and Practice 14(3), 143–164 (2009)
7. Brinkkemper, S.: Method Engineering: engineering of information systems development method and tools. Information and Software Technology 38(7) (1996)
8. Harmsen, A.F., Brinkkemper, J.N., Oei, J.L.H.: Situational Method Engineering for information Systems Project Approaches. In: nt. IFIP WG8. 1 Conference in CRIS Series: Methods and associated Tools for the Information Systems Life Cycle, vol. (A-55), pp. 169–194. North Holland (Pub.), Amsterdam (1994)
9. Brinkkemper, S., Saeki, M., Harmsen, F.: A method engineering language for the description of systems development methods (Extended abstract). In: Dittrich, K.R., Geppert, A., Norrie, M.C. (eds.) CAiSE 2001. LNCS, vol. 2068, p. 473. Springer, Heidelberg (2001)
10. Van Slooten, K., Hodes, B., Characterising, I.S.: development projects. In: Proceedings of the IFIP WG8.1 Conference on Method Engineering (1996)
11. Ralyté, J., Deneckere, R., Rolland, C.: Towards a Generic Model for Situational Method Engineering. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681. Springer, Heidelberg (2003)

12. Karlsson, F., Agerfalk, P.J.: Method configuration: adapting to situational characteristics while creating reusable assets. Information and Software Technology 46(9) (2004)
13. Wistrand, K., Karlsson, F.: Method components – rationale revealed. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 189–201. Springer, Heidelberg (2004)
14. Agerfalk, P.J.: Information systems actability: Understanding Information Techology as a Tool for Business Action and Communication. Doctoral dissertation. Dept. of Computer and Information Science, Linköping University (2003)
15. Henderson-Sellers, B.: Process meta-modelling and process construction: examples using the OPF. Ann. Software Engineering 14(1-4), 341–362 (2002)
16. Henderson-Sellers, B., Gonzalez-Perez, C., McBride, T.: A meta-model for assessable software development methodologies. Software Quality Journal 13(2) (2005)
17. Guzélian, G., Cauvet, C.: SO2M: Towards a Service-Oriented Approach for Method Engineering. In: The 2007 World Congress in Computer Science, Computer Engineering and Applied Computing, in the Proceedings of the International Conference IKE 2007, Las Vegas, Nevada, USA (2007)
18. Deneckère, R.: Approche d'extension de méthodes fondée sur l'utilisation de composants génériques, PhD thesis (in French), University of Paris 1-Sorbonne (2001)
19. Cossentino, M., Seidita, V.: Composition of a new process to meet agile needs using method engineering. In: Choren, R., Garcia, A., Lucena, C., Romanovsky, A. (eds.) SELMAS 2004. LNCS, vol. 3390, pp. 36–51. Springer, Heidelberg (2005)
20. Terracina, G., Garro, A., Ursino, D.: A multi-agent system for supporting the predition of protein structures. In: ICAE, vol. 11(3), pp. 256–280. IOS Press, Amsterdam (2004)
21. Method fragment definition, FIPA Document (2003), `http://www.fipa.org/activities/methodology.html` (accessed by November 2003)
22. Cossentino, M., Gaglio, S., Henderson-Sellers, B., Seidita, V.: A metamodelling approach for method fragment comparison, Proceedings of the 11th International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD), Luxembourg (2006)
23. Kornyshova, E., Deneckere, R.: A Framework for Comparing SME Approaches, Working paper, Centre de Recherche en Informatique, University of Paris 1 (2010)
24. Pohl, K., Böckle, G., Van der Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer, Heidelberg (2005)
25. Deneckère, R., Kornyshova, E.: Process line configuration: An indicator-based guidance of the intentional model MAP. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) BPMDS 2010 and EMMSAD 2010. Lecture Notes in Business Information Processing, vol. 50, pp. 327–339. Springer, Heidelberg (2010)
26. Rolland, C., Prakash, N., Benjamen, A.: A Multi-Model View of Process Modelling. In: Requirements Engineering, vol. 4, Springer-Verlag London Ltd., London (1999)
27. Roy, B.: Multicriteria Methodology for Decision Aiding, Dordrecht. Kluwer Academic Publishers, Dordrecht (1996)
28. Karlsson, J., Ryan, K.: A Cost–Value Approach for Prioritizing Requirements. IEEE Software (1997)
29. Rational Unified Process, Electronic Resource (2007), `http://www-306.ibm.com/software/awdtools/rup/` (accessed by June 2007)
30. Saaty, T.L.: The Analytic Hierarchy Process. McGraw-Hill, NY (1980)

Structural Aspects of Data Modeling Languages

Terry Halpin

LogicBlox, Australia and INTI International University, Malaysia
terry.halpin@logicblox.com

Abstract. A conceptual data model for an information system specifies the fact structures of interest as well as the constraints and derivation rules that apply to the business domain being modeled. The languages for specifying these models may be graphical or textual, and may be based upon approaches such as Entity Relationship modeling, class diagramming in the Unified Modeling Language, fact orientation (e.g. Object-Role Modeling), Semantic Web modeling (e.g. the Web Ontology Language), or deductive databases (e.g. datalog). Although sharing many aspects in common, these languages also differ in fundamental ways which impact not only how, but which, aspects of a business domain may be specified. This paper provides a logical analysis and critical comparison of how such modeling languages deal with three main structural aspects: the entity/value distinction; existential facts; and entity reference schemes. The analysis has practical implications for modeling within a specific language and for transforming between languages.

1 Introduction

A *conceptual data model* includes a conceptual schema (structure based on concepts that are intelligible to business users) as well as a population (set of instances that conform to the schema). A conceptual schema specifies the fact structures of interest as well as the business rules (constraints or derivation rules) that apply to the relevant business domain. Various languages are used by modelers to capture or query the data model. These languages may be graphical or textual.

In *attribute-based approaches* such as Entity Relationship modeling (*ER*) [2] and the class diagramming technique within the Unified Modeling Language (*UML*) [18]), facts may be instances of attributes (e.g. Person.isSmoker) or relationship/association types (e.g. Person drives Car). UML's Object Constraint Language (OCL) [19, 21] provides a textual means to express class diagrams as well as many additional rules.

In *fact-oriented modeling* approaches, such as Object-Role Modeling (*ORM*) [10], all facts are treated as instances of fact types, which are represented using typed, logical predicates (e.g. Person smokes, Person drives Car). Referential facts also involve existential quantification (e.g. **some** Country has CountryCode 'AU'). For a detailed coverage of ORM and comparisons with ER and UML see [13]. Overviews of fact-oriented modeling approaches, including history and research directions, may be found in [9, 11]. The Semantics of Business Vocabulary and Business Rules (SBVR) initiative [20] and the Object-Oriented Systems Modeling (OSM) approach [6] are also fact-based in their requirement for attribute-free constructs.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 428–442, 2011.

Declarative, logic-based languages are being increasingly used for data models that require rich support for logical derivation. The Web Ontology Language (*OWL*) [23], based on description logics, is designed to capture ontologies for the Semantic Web. Business intelligence tools and rule-based software are now widely used to perform predictive analytics over massive data sets and enforce complex business rules. This has led to a resurgence of interest in *datalog*, because of its powerful deductive database capability for processing complex rules, especially recursive rules [1].

Although sharing many aspects in common, these data modeling languages also differ in fundamental ways that impact not only how, but which, aspects of a business domain may be specified. This paper provides a logical analysis and critical comparison of how such modeling languages deal with three main structural aspects: the entity/value distinction; existential facts; and entity reference schemes. The analysis has practical implications for modeling within a specific language and for transforming between such modeling languages.

The rest of this paper is structured as follows. Section 2 discusses different ways in which modeling languages distinguish between entities and values, and the impact this has on modeling facts about them. Section 3 motivates the need for existential facts and the different ways (e.g. skolemization) in which these are supported (if at all) in the modeling languages. Section 4 briefly examines the relationship between skolemization and entity reference schemes. Section 5 summarizes the main contributions and outlines future research directions.

2 Entities and Values

In Chen's original ER model [2], an *entity* is defined as "a 'thing' which can be distinctly identified", a "relationship" is defined as "an association among entities", and information about entities or relationships is stored using attribute-value pairs in mathematical relations. For example, the *value* "AU" (an instance of the value set "CountryCode") may be used to represent the entity that is the country Australia. A typical, modern ER definition for the term "entity" is "a real-world object with an *independent existence*" (e.g. [3, p. 373], [5, p. 43]). Here an object may be physical (e.g. a person) or abstract (e.g. a course). Nowadays in ER modeling, the term "value" typically means a data value (instance of a value type based on a given datatype), such as a person's family name or a course code. One or more attributes or relationships of an entity are chosen to provide its primary identifier, which identifies the entity by mapping it (directly or indirectly) to its referencing value(s). In this paper, we use the term "entity" to mean an entity instance, not an entity type.

The above definitions for "entity" have issues. As a trivial issue, the world being modeled (the business domain of interest) need not be "real" in the normal sense of the word (e.g. consider a data model about fictional characters in the Harry Potter novels). As a substantive issue, the requirement of independent existence is debatable. In what sense does an entity such as a country exist independently? This notion is difficult to capture conceptually with any rigor. The motivation for the independent existence requirement seems to be to distinguish entities (e.g. countries) from attribute values (e.g. country codes), since attributes are always attributes of something and in that sense are not independent. In the ER approach, only entities and relationships

may have attributes or participate in other relationships. In practice, this seems to assume that we may record facts about entities and relationships, but not about values.

In UML, the terms "object", "class", and "data value" roughly correspond to "entity", "entity type", and "value" in ER. UML objects are assigned internal, object identifiers, but data values are not. There is no requirement in UML to provide value-based identifiers that are visible to human users. For practical data modeling however, value-based identification is typically needed to ensure that users are able to communicate about the entities of interest within a model, and to know when the same entity is referenced in different models. Hence UML data modelers typically use tools that extend UML with features such as key attributes, or they write the OCL needed to ensure value-based identification. Unlike ER entities, UML objects may include operational properties, but these operations are irrelevant to our discussion.

Consider a model in which each employee is identified by an employee number and also has exactly one honorific title (e.g. "Dr", "Mr" or "Ms"). A Barker ER diagram is shown for this in Fig. 1(a). Here Employee is modeled as an entity type with employee number and honorific as attributes. Employee numbers are used by humans in the business domain as the preferred identifiers for employees, so they are not hidden object identifiers in the UML sense, even if auto-generated. Fig. 1(c) shows a UML class diagram for this situation. The UML diagram omits the constraint that employee numbers are identifying, but this does not concern us here. If there is no need to talk about honorifics themselves, we believe that most ER and UML modelers would naturally model honorific as an attribute rather than as an entity type or class. An honorific instance (e.g. "Dr") is then conceived of as a value, not an entity.

In the real world, some honorifics have expansions (e.g. "Dr" expands to "Doctor" but "Ms" has no expansion), but assume initially that this is not of interest to the business domain. Now suppose that the business later discovers a need to display honorific expansions. In ER and UML, attributes can't have attributes or participate in relationships, so the honorific attribute needs to be remodeled as an entity type or class, and a binary relationship/association must now be used to model the former facts about employee honorifics, as shown in Fig. 1(b) and Fig. 1(d). Moreover, any code or user interface that accessed the former honorific attribute needs to be modified. An honorific instance is now conceived of as an entity (or UML object), not a value.

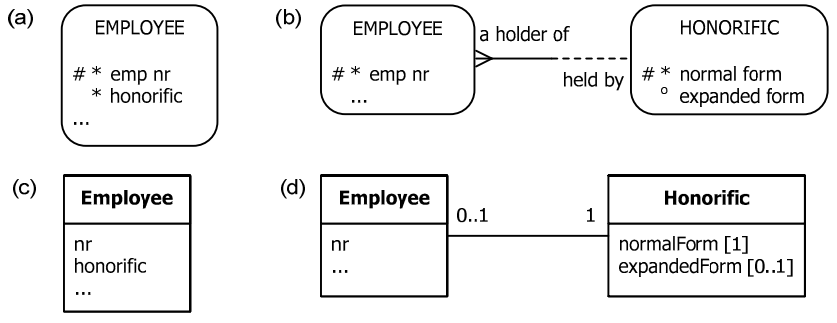


Fig. 1. Modeling employee honorifics in Barker ER (a), (b) and UML (c), (d)

Apart from the inconvenience of having to remodel already existing facts, changes of this nature have philosophical implications. If it was correct to initially treat an honorific instance as a value and also correct to finally model it as an entity, then it seems possible for a value to change to an entity. Moreover, in this case the change in an honorific's nature seems to be caused by the mere act of recording a fact about it (e.g. the fact that the Honorific "Dr" is short for the HonorificExpansion "Doctor"). This is somewhat reminiscent of Heisenberg's Uncertainty Principle, where the mere act of observing something necessarily changes it. However, *it seems implausible that a thing can change its nature (e.g. a value becomes an entity) simply because we want to talk about it*. Is there some way of drawing the entity/value distinction that does not force us into such seeming absurdities?

One extreme response might be to adopt a dynamic, relativistic theory where a thing is a value or entity only relative to a state of the business domain. So within a business domain, something is an entity at time *t* just in case the business wishes at time *t* to record facts about it (other than facts using it to reference another entity). However, this approach still has the semantic instability problem just described, and would seem to add considerable complexity to any underlying formalization.

An alternative solution that simply avoids such problems is provided by fact-oriented approaches such as ORM. In ORM a value may be defined as a *self-identifying constant* of a specified finite type, where the type name is typically informative (e.g. Honorific) or simply indicative of a conceptual datatype (e.g. CharacterString). So values can be verbalized by definite descriptions that simply include the lexical constant and a value type name (e.g. "the Honorific 'Dr'").

In contrast, an entity in ORM requires a *reference scheme* that includes at least one *referential relationship*. For example, the definite description "The Employee who has the employee number 2011" involves a specific binary relationship between the employee and the number. Moreover, entities typically change their state over time, so are not usually constant (unlike values). Hence in ORM it is impossible for a value to change to an entity.

ORM is *attribute-free*, so all facts are represented by relationships over one or more objects. In ORM, an object is the same as an *individual* in classical logic, so it can be an entity or value. Hence, in ORM *entities or values may appear in any position in a relationship*.

Fig. 2(a) shows an ORM schema and sample population for the ER example in Fig. 1(a). Entity types appear as named, solid, rounded rectangles, and value types appear as named, dashed, rounded rectangles. Relationship types are depicted using logical predicates which display as named, ordered sets of role boxes connected to the object types whose instances play those roles. An asserted fact type is either elementary or existential. A non-existential fact type is a set of one or more typed predicates, which may be unary, binary, or of higher arity. An elementary fact can't be rephrased as a conjunction of smaller facts with the same objects without information loss.

An injective relationship from an entity type to a value type that is used for entity identification is called a refmode predicate, and may be displayed in abbreviated form by enclosing the refmode in parenthesis below the entity type name. The bar over the first role of the Employee has Honorific predicate is a uniqueness constraint (each employee has at most one honorific). The solid dot on the role connector is a mandatory role constraint (each employee has some honorific).

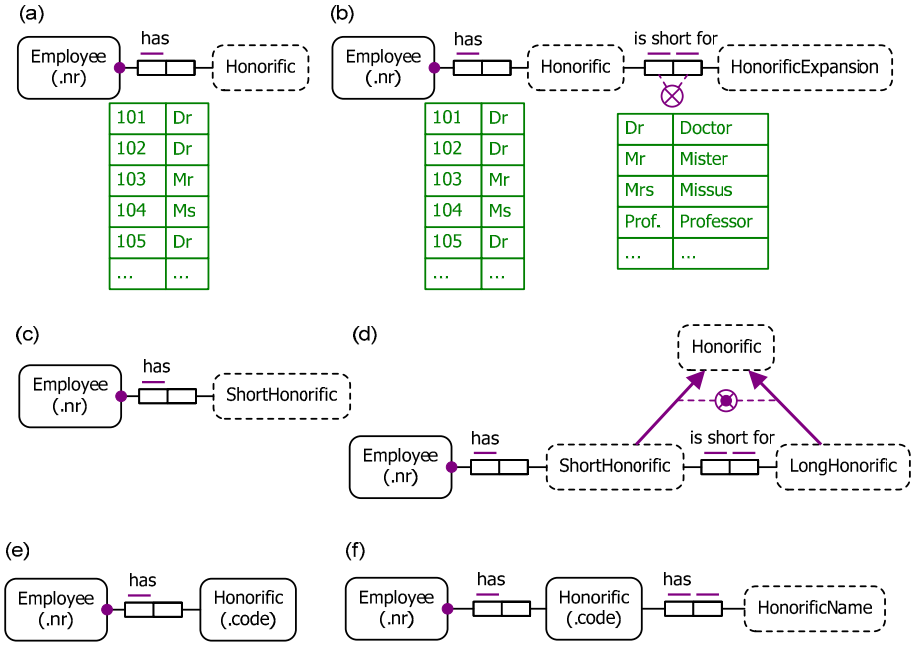


Fig. 2. Modeling employee honorifics in ORM

Fig. 2(b) adds the optional, 1:1 relationship type Honorific is short for HonorificExpansion. Notice that this addition has no impact on the original model in Fig. 2(a). This a simple illustration of the greater *semantic stability* enabled by fact-orientation in comparison with attribute-based approaches. Facts may be added about any kind of object (entity or value) without impacting the existing model. The circled cross denotes an exclusion constraint (no honorific is an honorific expansion).

The models in Fig. 2(a) and Fig. 2(b) use "honorific" in a restricted sense to mean the usual short title applied to a person's name (e.g. "Dr"). If "honorific" is used in the business domain to include longer titles (e.g. "Doctor"), then the type names should be adjusted accordingly (e.g. "ShortHonorific" and "LongHonorific"). Fig. 2(c) would then be used as the initial schema, and the lower part of Fig. 2(d) could be used as the expanded schema. If the business wishes to talk about honorifics in general, then the supertype Honorific may be introduced as in Fig. 2(d). The circled, dotted cross between the subtyping connections denotes an exclusive-or constraint (Honorific is partitioned into ShortHonorific and LongHonorific).

The ORM models in Figures 2(a)-(d) conceive of honorifics as simple labels (and hence values). However, suppose the modeler feels that "Dr" and "Doctor" are just different representations of the same honorific. With this understanding, an honorific is an entity (e.g. a personal status concept), not a value. Fig. 2(e) and Fig. 2(f) show one way to model this in ORM. Honorific is now an entity type. The short label for an honorific is called an honorific code, and the longer label is called an honorific name.

In practice, different people sometimes assign different meanings to the same term. Hence whether a "thing" is conceived of as an entity or value is sometimes relative to

the user. The honorifics example is a borderline case, where it is not unreasonable to take either view with respect to the entity/value status of honorifics. However, once the meaning of a term within the business domain is determined, it should be modeled consistently, otherwise the benefits of semantic stability are lost. With this understanding, *within a given model, no value may change to an entity, or vice versa*.

In most cases, an entity can be intuitively distinguished from a value simply by noting that it is not lexical in nature, or that it can be referenced by labels in more than one way, or that it can change over time. For example, a country can be referenced by its relationship of having a country code (e.g. 'AU') or a country name (e.g. 'Australia'), and it evolves over time. One often sees ER or UML models where gender is modeled as an attribute; but a gender is an abstract concept that can be referenced by its relationship to a gender code (e.g. 'M') or a gender name (e.g. 'Male'). Even though a gender does not change over time, it is clearly not a label like a gender code. By modeling gender and honorific details for persons as relationships between Person and Gender and Honorific object types, we can add the optional, functional relationship type Honorific is restricted to Gender, populate it with data (e.g. the honorific 'Mr' is restricted to the gender with code 'M'), and add the join subset constraint **If a** Person has **an** Honorific **that** is restricted to **some** Gender **then that** Person is of **that** Gender [14].

In practice, commercial versions of ER are typically restricted to single-valued attributes. This restriction, combined with the inability to model facts about attribute values, leads to further semantic instability. For example, suppose we began with the ER model of Fig. 1(a), and later needed to cater for the possibility of recording multiple honorifics for the same employee (e.g. 'Prof.' and 'Dr'). In Barker ER, we would need to replace the honorific attribute with an *m:n* relationship from Employee to an Honorific entity type. In ORM, we simply change the uniqueness constraint on the employee-honorific relationship to be spanning. UML can cater for this change by using a multivalued honorific attribute (change its multiplicity in Fig. 1(c) to [*]), but in practice multivalued attributes can be more trouble than they are worth ([13], p. 356).

We now discuss entities and values in logic-based languages, starting with OWL 2, the latest version of the Web Ontology Language. An overview of OWL's structure is shown in Fig. 3, which is adapted from the official specification [26].

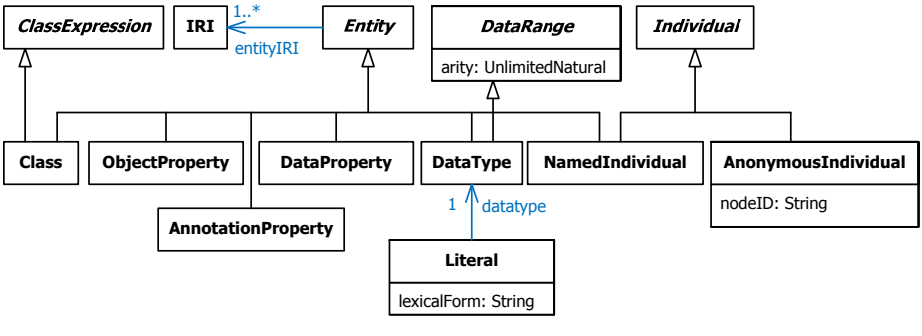


Fig. 3. UML class diagram of basic OWL concepts, adapted from [26]

OWL identifies entities by Internationalized Resource Identifiers (IRIs) [4], but unlike some approaches, OWL does not adopt the Unique Name Assumption, so the same entity may be assigned different IRIs, even within the same document. Hence the multiplicity constraint on entityIRI is 1..* (1 or more), not 1 as specified in [26].

As can be seen from Fig. 3, OWL uses the term "entity" in a much broader sense than we have been considering. For example, a class is itself an entity, and in OWL Full a class can even be an instance of itself, inviting Russell's paradox.

OWL *properties* are binary predicates, and their instantiations are treated as entities, similar to instances of class associations in UML and, to some extent, objectification in ORM. OWL *individuals* are either named or anonymous. Anonymous individuals are discussed in the next section. *Named individuals* are typical of the entities we discussed earlier, except that they are identified by an IRI. *Literals* roughly correspond to what we have been calling values. A literal has a lexical form (quoted string, for which a language tag may optionally be specified) and a datatype, which may be hidden if it is rdf:PlainLiteral (see pp. 37-39 of [26] for details).

Note that *OWL literals are not treated as individuals*, and so OWL differs from classical logic in this respect, where, for example, you can use the individual constant "AU" to refer to the individual character string inside the quotes. *Object properties* are binary predicates that relate individuals to individuals. *Data properties* are binary predicates that relate individuals to literals. For example, if within the local document "Einstein" and "Germany" serve as IRIs, then we can declare Einstein's birth country and name in Manchester Syntax [25] thus:

```
ObjectProperty: wasBornIn
DataProperty: hasName
Individual: Einstein
   Facts: wasBornIn Germany, hasName "Albert Einstein"^^xsd:string
```

OWL's distinction between entities and literals seems to be taken very loosely in practice. For example, the official OWL 2 Primer cites as examples of data values "a person's birth date, his age, his email address etc." [23, p. 21], giving the following example of a data property stating that John's age is 51:

```
Individual: John
   Facts: hasAge "51"^^xsd:integer
```

In conceptual data modeling, an age is a duration in time with a unit (e.g. years), so an age is an entity, not a data value. Similarly, a date is a 24 hour period (anchored duration in time), so is an entity, unlike a date string, which is a value. An e-mail address may be conceived as a value, though a home address could be thought of as either a physical location (an entity) or a value (possibly structured).

OWL is built on top of the Resource Description Framework (RDF), so OWL facts are expressed as subject-predicate-object triples, and *the subjects of OWL facts must be individuals, not literals*. So OWL is unable to model fact types of the form *A R B*, where *A* is a value type, such as ShortHonorific is short for LongHonorific in Fig. 2(b), or the ORM synonym fact types in Fig. 4. Here, "Word" means English word, and its instances are represented by character strings, just as they would typically be stored in a relational database. Of course, we could model words in OWL by treating them as entities, but it seems subconceptual to require an IRI in order to talk about a word.

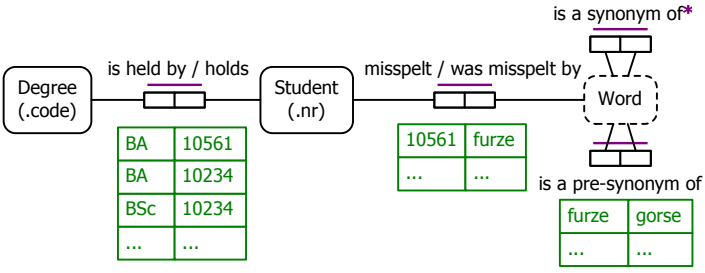


Fig. 4. An ORM model about students and word knowledge

For binary fact types in ORM, a slash may be used to separate forward and inverse predicate readings. In Fig. 4, the student degree fact type has two readings: Degree is held by Student; Student holds Degree. The fact type used to record student misspellings also has two readings: Student misspelt Word; Word was misspelt by Student. As well as supporting natural communication by allowing facts to be expressed in different ways, inverse readings often facilitate more natural verbalization of rules that involve navigation over paths that traverse multiple fact types. Barker ER supports forward and inverse relationship readings. UML supports only one association reading per association, but allows navigation in different directions across an association by use of role names. OWL supports inverses of object properties. For example, both predicates for the student-degree fact type in Fig. 4 may be declared in Manchester Syntax thus:

```
ObjectProperty: isHeldByStudent
  InverseOf: holdsDegree
```

However, while the student-misspelling fact type may be declared as a data property using the "misspelt" predicate, its inverse predicate "wasMisspeltBy" cannot be declared at all because its subject is a literal type. The only way around these problems in OWL is to remodel Word as an entity type. Even if it is reasonable to conceive of a word as an entity not a value, there are many cases where such a workaround seems unnatural. Most modelers consider names to be values, not entities, and the earlier example of using a data property to record Einstein's name is typical in OWL. However, if we do this, we cannot express the inverse relationship that would be modeled in ORM using PersonName is of Person.

Suppose we initially model names or codes etc. as values, but then wish to talk about them (e.g. record their origin, meaning, purpose, or length). Do they now suddenly become entities? We think not. Clearly there is a difference between a country and a country name or country code. You can live in a country, but you can't live in a country code. However, there are different stances one might take with respect to the nature of values themselves, as used in conceptual modeling.

Consider the country code "us" and the pronoun "us". Are these identical values? If values are simply untyped, lexical constants, then the answer is Yes: it's the same value being used for two different purposes. The value types CountryCode and Pronoun are then understood (implicitly or explicitly) to be finite, overlapping subtypes of a datatype such as CharacterString. However, suppose we populate the fact type Pronoun is plural with "us". If the pronoun "us" = the country code "us" then the principle of substitutivity of identicals entails that the country code "us" is plural, which is nonsense.

The only way to make sense of this untyped constant approach is to implicitly expand all predicate readings to include types (e.g. rewrite "is plural" as "is a plural pronoun" or "pronoun:isPlural").

If instead we consider the notion of a value to intrinsically include a specific purpose (represented by a type name), then the answer is No: pronouns and country codes are different kinds of things. This second position treats values as strongly typed constants, and a distinction may be drawn between strong identity and lexical equality when comparing values. So the pronoun "us" and the country code "us" are strictly non-identical while still being lexically equal. With this approach, value types may have a datatype but are not subtypes of datatypes. It is still possible to have overlapping value types (e.g. CountryCode overlaps with TwoLetterCode).

We now briefly discuss how entities and values are treated in *datalog*, a logic-based language that is becoming widely used in business analytics and rule-based systems. Like Prolog, datalog is a logic programming language. Unlike Prolog, it is purely declarative and has guaranteed decidability. As datalog is based on classical logic, it allows predicates to be applied to individuals that may be entities or values, with no restriction on where they may appear in predications. Datalog supports powerful inferencing capabilities, allowing complex rules (including recursive rules) to be elegantly formulated and efficiently processed. For illustration, we use Datalog^{LB}, an extended form of typed datalog. A theoretical discussion of datalog may be found in [1], and an overview of mapping ORM to Datalog^{LB} may be found in [12].

Let's see how the misspelling and synonymy fact types in Fig. 4 might be declared and populated in Datalog^{LB}. Derived fact types in ORM are indicated with an asterisk. The synonymy fact type may be derived using the following FORML [14] rule: Word$_1$ is a synonym of Word$_2$ **iff** Word$_1$ is a pre-synonym of Word$_2$ **or** Word$_2$ is a pre-synonym of Word$_1$.

ORM entity types and value types help understand and visualize models, especially how things are connected. When mapping an ORM model to a model in another language, you may leave value types implicit (e.g. by informally including the semantics in a predicate reading such as "hasCountryCode"). In the following Datalog^{LB} program, the Word type is implicitly treated as a subtype of string. The declarations constrain the types of the predicate arguments (read "->" as "implies", and "," as "and"). The derivation rules derive the inverse misspelling predicate and the synonymy predicate (read "<-" as "if" and ";" as "or"). Some data and a sample query are included. If you wish to make Word an explicit value type, declare its type as string, replace string by Word in the other declarations, and add +Word("furze"), +Word("gorse") to the data.

```
//declarations
Student(s), hasStudentNr(s:n) -> uint[32](n).
misspelt(s, w) -> Student(s), string(w).
isPreSynonymOf(w1,w2) -> string(w1), string(w2).
//rules
wasMisspeltBy(w,s) <- misspelt(s, w).
isSynonymOf(w1,w2) <- isPreSynonymOf(w1,w2);  isPreSynonymOf(w2,w1).
//data
+Student(10561), +misspelt(10561, "furze" ), +isPreSynonymOf("furze","gorse").

sample query:        isSynonymOf   ⇨  furze, gorse
                                       gorse, furze
```

3 Existential Facts

Conceptually, a fact base (as distinct from constraints or rules) may be expressed as a set of elementary or existential facts. An elementary fact is an atomic predication over named individuals (e.g. Einstein is male, Einstein was born in Germany). An *existential fact* asserts the existence an individual, typically to predicate over it (e.g. some person is male, some person was born in Germany). To facilitate a first-order formalization, we do not treat existence as a predicate. Most but not all logicians agree that if existence is treated as a predicate, it must be construed as a second-order predicate. For further discussion on "exists" as a predicate, see [17].

In typical relational database applications, simple existential facts like the examples above are never stored, even though they may be implied. If we store the fact that the politician Obama is the president of the USA, we can infer that some politician is the president of the USA. But knowing that some politician is the president of the USA doesn't enable us to infer who that is. On the surface then, it may appear that there is little reason for data models to even be concerned with existential facts.

However, there are cases where support for existential facts is vital. One case is *data exchange* between different schemas that are not logically equivalent, even when supplemented by conservative extension derivation rules (for a formalization of ORM schema equivalence under conservative extension see [8]). Rules that map data between the models may be set out as tuple-generating dependencies of the form $\forall \boldsymbol{x},\boldsymbol{y}[\Phi(\boldsymbol{x},\boldsymbol{y}) \rightarrow \exists \boldsymbol{z} \Psi(\boldsymbol{x},\boldsymbol{z})]$, where $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ are variable lists, and $\Phi(\boldsymbol{x},\boldsymbol{y})$ and $\Psi(\boldsymbol{x},\boldsymbol{z})$ are conjunction of atoms from the source and target schema respectively (e.g., see [7], [16]). For example, suppose both the source and target model information about scientists, but only the second records their birth countries, and has a constraint that each scientist has a birth country, i.e. $\forall x[\text{Scientist}(x) \rightarrow \exists y \text{ wasBornInCountry}(x,y)]$. To map details about Einstein from the first to the second model, a *skolem constant* is introduced there to denote Einstein's birth country. Queries that include the birth country will now return the null set, but queries that project only on non-skolem attributes work fine.

A related application of existential facts is support for *updating views* that involve joins. Suppose the database includes the base relation scheme parentOf(parent, child) as well as the view grandparentOf(grandparent, grandchild) derived from the rule $\forall x,y$ [grandparentOf$(x,y) \leftarrow \exists z$(parentOf$(x,z)$ & parentOf(z,y)]. In a normal relational database, if we attempt to insert the fact grandparentOf(Bernie, Selena) into the view, this will be rejected, since the update can't be translated into updates on the base parentOf relation. Adding the tuples parentOf(Bernie, null) and parentOf(null, Selena) won't help, even if allowed, because nulls never match (comparisons with null return unknown). For a discussion of this example in SQL see [13, p. 649].

However, if instead we use a logic-based database that supports skolem terms, we can accept the view update simply by using the same skolem term for the intermediate unknown parent. From an ORM perspective, the grandparenthood fact type is now *semi-derived*, since some of its instances may be simply asserted and other instances may be derived from parenthood facts.

Both OWL and Datalog^{LB} support existential facts, though there are some differences in their approaches. In OWL, individuals that are referenced by skolem constants are called *anonymous individuals*, and correspond to blank nodes in RDF (see

section 2.3 of [22]). A skolem constant itself (e.g. _:a or _:b) is called a nodeId (see Fig. 3), and may be read as "something" if it's the only skolem term in the statement; otherwise the reading should include the id name (e.g. "some a" or "some b"). In OWL, a skolem constant is simply an arbitrary constant that replaces an existential quantification within the scope of the current statement.

In an effort to support the AAA assumption (Anybody can say Anything about Anything), OWL places few restrictions on use of anonymous individuals. For example, you can simply assert that some god exists, and that some woman is the prime minister of Australia (without knowing that it's Julia Gillard). The following OWL statements in Turtle syntax do this using local nodeIds for anonymous individuals. Although these are legal in OWL, some OWL tools (e.g. Protégé) do not support use of nodeIds in this way. For a detailed overview of OWL syntaxes, see [23].

```
_:x  rdf:type  :God.
_:y  rdf:type :Woman ; :isPrimeMinisterOf :Australia.
```

Even within the OWL community, there are some who see little use for asserted existential facts, except for cases where blank nodes simply serve the purpose of joining facts, as in the RDF graphs shown in Fig. 5. By introducing the blank node "_:c" for "some city" in Fig. 5(a), we can assert that Einstein was born in a city that has a population of 121650, without knowing which city it is. We delay discussion of Fig. 5(b) till the next section, as it bears on the topic of reference schemes. For a somewhat humorous debate on the worth or otherwise of skolem terms, see the "OWL 2 Far" panel discussion segment between Stefan Decker and Ian Horrocks [15].

In classical datalog, existential facts are allowed only in the body of a rule, and a *rule* is an expression of the following form, where the *head* predicate q has as argument an ordered list of individual terms $\tau_1, \ldots \tau_n$ ($n \geq 0$), each variable of which must occur in at least one argument of the *body* predicates $p_1 \ldots p_m$ ($m \geq 0$).

$$q(\tau_1, \ldots, \tau_n) \leftarrow p_1(x_1, \ldots), \ldots, p_m(y_1, \ldots).$$

In classical datalog, a rule is treated as shorthand for a formula where the head variables are universally quantified at the top level, and any other variables introduced in the body are existentially quantified, with the existential quantifiers placed at the start of the body [1, p. 279]. For example, the datalog rule grandparentOf(x, y) ← parentOf(x, z), parentOf(z, y) is interpreted as shorthand for the following predicate logic formula: $\forall x \forall y$[grandparentOf(x, y) ← $\exists z$(parentOf(x, z) & parentOf(z, y))].

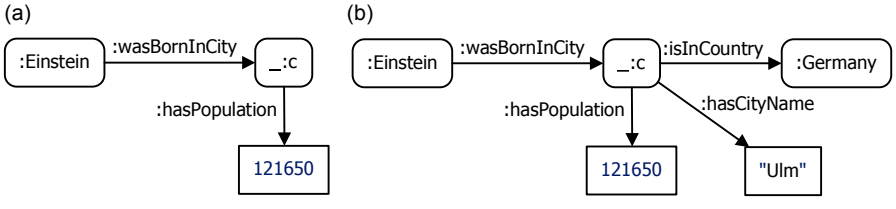


Fig. 5. RDF graphs using a blank node to assert the existence of some city

If an existentially quantified variable appears more than once in the body, a named, individual variable is used (e.g. "*z*" in the above example). If it appears only once, an anonymous variable "_" may be used, since we don't need to refer to it elsewhere. For example, consider the rule Parent(p) ← Person(p), parentOf(p,_).

What OWL calls blank nodes are thus handled in datalog using named or anonymous variables, where the variables are understood to be existentially quantified. If you mentally ignore the implicit existential quantifiers, you may think of these variables as skolem constants, except that multiple anonymous variables in the same rule must be treated as distinct (i.e. they may or may not be equal). For example, consider the datalog rule CarDrivingParent(p) ← Person(p), parentOf(p,_), drivesCar(p,_). In OWL, we could distinguish the anonymous variables by using different nodeIds (e.g. _:x, _:y).

While classical datalog allows existential facts only in rule bodies, there are good reasons to allow them in rule heads, such as supporting the data exchange and view updatability cases mentioned earlier. Support for head existentials in just one area in which Datalog^{LB} extends classical datalog, and in the next section we discuss an application of this feature that is conceptually linked to the notion of reference schemes.

4 Reference Schemes and Head Existentials

One common use of blank nodes in OWL is to model entities that have a composite reference scheme. For example, if cities can be identified just by their name and country then the RDF graph in Fig. 5(b) could be used to model the object property isIn-Country and the data property hasCityname, which could then be constrained as mandatory, functional, key properties for City to provide a natural, value-based identification scheme. In practice, city names might not be unique to their country, but they are usually unique to their state.

Fig. 6 shows an ORM model in which states are identified by combining their state code and country. A uniqueness constraint shown with a double-bar indicates its use for a preferred reference scheme. Countries are identified simply by their country code (this reference scheme is shown in expanded form for discussion purposes). Finally, head politicians (e.g. presidents or prime ministers) are identified by the country that they head. With this schema we could record that fact that the government head of Australia was born in Wales, even if we don't know who he/she is (currently, it's Julia Gillard).

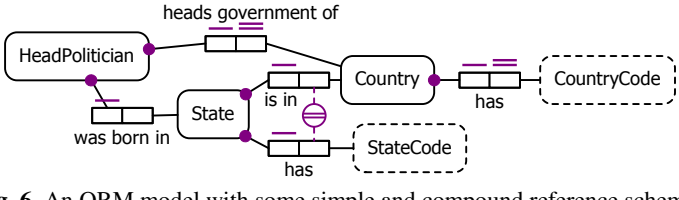


Fig. 6. An ORM model with some simple and compound reference schemes

In OWL, a head politician would be modeled as a blank node, and so would a state unless we have a natural IRI for it. For most states, a name based IRI could be used if known (e.g. :WashingtonState and :WestAustralia both have state code "WA"), but some states do have the same name, so this doesn't always work. Country names are

identifying, so countries would typically be identified by an IRI (e.g. :Australia). However, suppose that we want to talk about a country with country code "AU", but don't know its name. It would be strange to use an IRI such as :AU, so unless we are able to base an IRI on some Website fragment dealing with countries, we could then choose a blank node for countries as well. In that situation, a population of the model in Fig. 6 would include values for country codes and state codes, but all the other entities (in the normal sense of the word) would effectively be existentially asserted using reference predicates to provide definite descriptions that relate them to these values.

IRIs are essentially scoped, individual constants that are identifying within their namespace. If we don't have an IRI for an entity, in order to talk about it we must provide a definite description for it, and this always involves at least one reference predicate. The most general form of reference scheme is disjunctive reference, where each instance of an entity type is ultimately 1:1 mapped onto one or more values via reference predicates [13, pp. 187-188].

Fig. 7 shows a much simplified fragment of an ORM model to automatically generate verbalizations of ORM constraints. For example, the uniqueness constraint on the modality fact type has a negative verbalization that renders as "**It is impossible that some** Constraint has **more than one** Modality". The components of the verbalization (only the modal text part shown here) can all be derived from properties of the constraint.

In Datalog^{LB}, once the shaded predicate is declared as a skolem predicate and the verbalization's storage structure is declared as ScalableSparse, the fact type for the modal text can be derived using rules that existentially quantify the verbalization in the rule head, e.g. NegativeVerbalization(v), hasNegativeVerbalization[c]=v, hasModalText[v]="It is impossible that " <- hasModality[c]= "Alethic". This has the form $\forall c\ (\exists v \Phi vc \leftarrow \Psi c)$.

Currently, to generate the datalog code from ORM, the constraint verbalization entity type must be assigned an autogenerated id, which is used as a type specific skolem constant. Conceptually, the situation may be viewed as analogous to the head of government reference scheme in Fig. 6, where a definite description such as "the negative verbalization of the constraint with constraint number *n*" suffices. The autogenerated verbalization id may then be viewed as an implementation issue rather than as part of the pure conceptual model, allowing the conceptually preferred reference scheme to then be indicated by using a double-bar for the uniqueness constraint on Constraint's role in the skolem predicate.

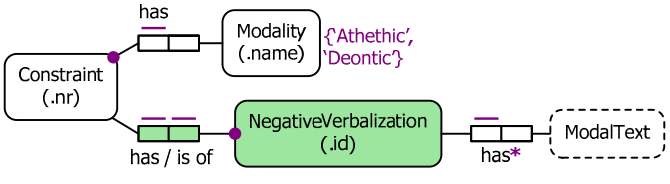


Fig. 7. A simplified ORM schema fragment involving skolemization

Even simple reference schemes such as the refmode predicate used to identify countries in Fig. 6 may be viewed as involving existential facts. As formalized in [8], the fact entry +Country("AU") asserts that there exists some country that has the country

code “AU”. This existential fact, when combined with the injective nature of the refmode predicate, licenses use of definite descriptions such as “the country that has country code “AU” for identifying entities. Viewed in this light, all data modeling approaches make use of existential facts, even though the approaches differ in the range of such facts that can be expressed and where they may appear in rules.

5 Conclusion

Although terms like “entity” and ‘value” are often used in the data modeling community, they may have different meanings in different modeling approaches. This paper reviewed these notions within different modeling languages, and opted for a semantically stable approach that draws the entity/value distinction on fundamental representational grounds rather than subjective and possibly changing viewpoints on what features one wishes to record facts about. Although the semantic instability of attribute-based approaches like ER and UML is well known, in this paper we showed that this semantic instability problem relates more fundamentally to an unwillingness to allow values to be subjects of facts. Hence, OWL also suffers from this instability.

The paper also provided a motivation for existential fact support, discussed some different ways in which this is provided in logic-based languages, and examined some connections between skolemization and reference schemes. Although languages like OWL and Datalog^{LB} provide basic support for these features, more work needs to be done to provide a comprehensive and purely conceptual approach that can be mapped to such languages for execution. Understanding the different ways in which modeling approaches deal with entity/value distinctions and existential facts is important not only for modeling within a given approach but for transforming between approaches.

Owing to space considerations, the coverage of values focused mainly on string-based representations, but even within this limited scope there is room for further analysis. For example, a simple definition of a lexical value is “something that you can write down”, but you can never write down a character string, only an occurrence of a representation of one. A full analysis of the entity/value distinction needs to embrace other kinds of data values (e.g. numeric and temporal), and properly account for unit-based reference. Different positions can also be taken on whether values can have conceptual structure. For example, is a person name composed of a given name and family name an entity or a “structured value”? As ongoing research not discussed here we are also refining the conceptual presentation of disjunctive reference schemes involving a partition of 1:1 predicates, as well as related subtyping alternatives.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. Chen, P.P.: The entity-relationship model—towards a unified view of data. ACM Transactions on Database Systems 1(1), 9–36 (1976), http://csc.lsu.edu/news/erd.pdf
3. Connolly, T., Begg, C.: Database Systems, 5th edn. Pearson Education, Boston (2010)

4. Duerst, M., Suignard, M.: RFC 3987: Internationalized Resource Identifiers (IRIs). IETF (January 2005), http://www.ietf.org/rfc/rfc3987.txt
5. Elmasri, R., Navathe, S.: Fundamentals of Database Systems, 2nd edn. Addison-Wesley, Menlo Park (1994)
6. Embley, D.: Object Database Management. Addison-Wesley, Reading (1998)
7. Green, T., Karvounarakis, G., Ives, Z., Tannen, V.: Update Exchange with Mappings and Provenance. Technical Report MS-CIS-07-26, Dept. of Computer and Information Science, University of Pennsylvania (2007)
8. Halpin, T.: A Logical Analysis of Information Systems: static aspects of the data-oriented perspective. Doctoral dissertation, University of Queensland (1989), http://www.orm.net/Halpin_PhDthesis.pdf
9. 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)
10. Halpin, T.: Object-Role Modeling. In: Liu, L., Tamer Ozsu, M. (eds.) Encyclopedia of Database Systems. Springer, Berlin (2009)
11. Halpin, T.: Object-Role Modeling: Principles and Benefits. International Journal of Information Systems Modeling and Design 1(1), 32–54 (2010)
12. Halpin, T., Curland, M., Stirewalt, K., Viswanath, N., McGill, M., Beck, S.: Mapping ORM to datalog: An overview. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6428, pp. 504–513. Springer, Heidelberg (2010)
13. Halpin, T., Morgan, T.: Information Modeling and Relational Databases, 2nd edn. Morgan Kaufmann, San Francisco (2008)
14. Halpin, T., Wijbenga, J.P.: FORML 2. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) BPMDS 2010 and EMMSAD 2010. Lecture Notes in Business Information Processing, vol. 50, pp. 247–260. Springer, Heidelberg (2010)
15. ISWC 2008 Panel discussion: An OWL 2 Far? (2008), http://videolectures.net/iswc08_panel_schneider_owl/
16. Kolaitis, P.: Schema Mappings, Data Exchange, and Metadata Management. In: PODS 2005, Baltimore, Maryland. ACM, New York (2005), 1-59593-062-0/05/06
17. Miller, B.: Existence. Stanford Encylopedia of Philosophy (2002), http://plato.stanford.edu/entries/existence/
18. Object Management Group: UML 2.0 Superstructure Specification (2003), http://www.omg.org/uml
19. Object Management Group: UML OCL 2.0 Specification (2005), http://www.omg.org/docs/ptc/05-06-06.pdf
20. Object Management Group: Semantics of Business Vocabulary and Business Rules, SBVR (2008), http://www.omg.org/spec/SBVR/1.0/
21. Warmer, J., Kleppe, A.: The Object Constraint Language, 2nd edn. Addison-Wesley, Reading (2003)
22. W3C: RDF Primer (2004), http://www.w3.org/TR/rdf-primer/
23. W3C: OWL 2 Web Ontology Language: Primer (2009), http://www.w3.org/TR/owl2-primer/
24. W3C: OWL 2 Web Ontology Language: Direct Semantics (2009), http://www.w3.org/TR/owl2-direct-semantics/
25. W3C: OWL 2 Web Ontology Language Manchester Syntax (2009), http://www.w3.org/TR/owl2-manchester-syntax/
26. W3C: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (2009), http://www.w3.org/TR/owl2-syntax/

Characterizing Business Rules for Practical Information Systems

Andrew Carver and Tony Morgan

INTI International University, Malaysia
andy.carver@yahoo.com, goldbasilisk@live.com

Abstract. The recognition of business rules as an important element of modern information systems has led to various proposals for business rule categorization schemes. In particular, a recent business rule standards proposal, the OMG standard for the Semantics of Business Vocabularies and Business Rules (SBVR) distinguishes between major categories of rule using a scheme derived from modal logic, based on alethic and deontic modalities. This paper examines some of the claims made for this categorization scheme in terms of the relationship with generally accepted logical systems, and identifies a number of problem areas. It further assesses the value of this modal logic classification scheme in the development and maintenance of information systems. Planned future work will look at an alternative scheme for practical categorization of business rules.

1 Introduction

Broadly speaking, business rules help to define the way that a business intends to operate. This can include discouraging or preventing undesirable activities, eliminating or correcting bad data, and so on. (Of course, rules can also be expressed in a positive way in terms of promoting desirable activities, etc.). The importance of business rules as first-class citizens in business information models was first described in the 1990s with the pioneering work of the Business Rules Group (BRG) [9]. The basic concepts have subsequently been refined and extended and documented in detail by several authors [1, 2, 3].

More recently, there have also been several initiatives aimed at defining standards relating to business rules. Potential advantages of standardized approaches include:

- improving industry understanding of business rules, through the clear definition of common core features;
- reducing the need for proprietary tools, languages, data structures, etc. that are limited to a single vendor;
- simplifying business analysis by providing a means of identifying patterns that are regularly encountered in business scenarios;
- reducing the cost of software development by encouraging common design techniques, community software libraries, etc.;
- facilitating the accurate transfer of business logic between systems and between organizations.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 443–452, 2011.

Examples of such initiatives include the OMG standards for the Semantics of Business Vocabularies and Business Rules (SBVR) [7] and Production Rule Representation (PRR) [8], and the W3C family of standards for a Rule Interchange Format (RIF) [6]. Another initiative, RuleML [5] is not a standard but is an organization that has influenced and made contributions to other standards, including PRR and RIF, but not SBVR.

A persistent interest has been the desire to classify rules into a number of categories. For example, the original BRG paper specified that each business rule must be one of the following:

- a structural assertion;
- an action assertion;
- a derivation.

Several of the contributors to the BRG paper subsequently also became contributors to SBVR, and their contributions modified and extended the original classification scheme to the significantly more complex scheme presented in SBVR. A major new feature introduced in SBVR is the linkage of the proposed categorization scheme to modal logics (alethic and deontic modalities) as a means of providing a formal grounding.

One aspect not addressed in SBVR is a clear explanation of how the categorization scheme assists in the practical aspects of building and maintaining working information systems. For example, if one correctly identifies a rule as belonging to Category A, how does that knowledge help to design or build an information system that incorporates the rule? As another example, if a rule originally implemented as Category A subsequently turns out to be in Category B, what changes would need to be made in an information system that incorporates the rule?

In the remainder of this paper we focus mainly on the relationship between business rules (i.e., rules expressed from a non-technical business perspective) and the way in which those rules could impact the design or construction of a practical information system. Although we are thinking primarily of modern computer-based information systems, most of the points we cover in the next section would be equally applicable in purely manual information systems of the kind used before the 1950s.

In Section 2, we examine the classification scheme proposed in SBVR; in particular, its use of alethic and deontic modal logics. In section 3 we consider further the impact of the way that rules are classified on the way that they are treated in practical information systems: typically (but not necessarily) systems that involve automation.

2 Rule Classification Based on Modal Logic

2.1 Reality and Information Systems

In this section we examine the classification scheme used in SBVR, which is based on concepts drawn from modal logics. SBVR focuses on two modalities: alethic and deontic. Details of these can be found in several reference sources such as [10]. Space here permits only a brief explanation, which will inevitably gloss over much detail,

but an appreciation of certain key points is an essential prerequisite to an understanding of the SBVR categorization scheme.

We will also need to distinguish between states of affairs as they exist in the real world, and the way that those same states of affairs are represented in an information system. In this paper we use "real world", "reality", etc. to mean the actual world of business activity, as regularly experienced every day in all kinds of organizations and in all kinds of locations. This world is governed both by what we might call "laws of nature" and by the way that organizations choose to conduct their affairs. Much information and activity that exists in the real world is ignored from a business perspective, because it has no relevance to the business of the organization (best friend? color of socks? choice of breakfast?). However, a business may choose to encourage or discourage certain behaviors and information (in various extents) in order to further its aims.

In contrast to the natural, information-rich environment of the real world, an information system is much more circumscribed. Information systems (whether manual or automated) are artifacts. They are typically designed to serve some particular purpose or purposes, and their intended goal is likely to influence their structure. They will normally only include information relevant to their goal, because unnecessary information would increase cost, development time, etc., and decrease performance, maintainability, etc. without adding any benefit to the organization. The information content of the system can be regarded as a selective abstract of the real world information that exists in the business. The reasons for explicitly creating an information system to cover this subset of the real word include ease of access, persistence of records, computation, etc.; facilities that are important to the operation of the business. It is important to note that the information content of a system (automated or otherwise) is not inherently governed by "laws of nature". Any such "laws" are manifested in an information system only if the business explicitly chooses to address them. For example, a person has only one birth date in reality, but could have any number in a (badly designed) information system.

2.2 Alethic Modality

Alethic modal logic deals with notions such as necessity (something is always the case) and possibility (something may be the case) and so on. In fact, from necessity along with some judicious negation, we can derive five possible statuses.

- Necessity: It is necessary that X
- Possibility: It is possible that X (it is not necessary that not X)
- Contingency: It is not necessary that X and not necessary that not X
- Impossibility: It is impossible that X (it is necessary that not X)
- Non-Necessity: It is not necessary that X

The last two of these are not directly represented as concepts in SBVR, but this is not a problem since they can be represented alternatively. For example, Impossibility can always be represented as a Necessity, as shown above. (Although SBVR does not include Impossibility as a concept, it does include Impossibility Statements, which are treated in the same way as Necessity Statements, for the reasons explained above).

Given the key role of Necessity in alethic modality, it is important to understand the commitment that is being made. If something is categorized as necessarily true, then it is true in ALL possible worlds. That would include all of history (whether recorded or not), the present, and any situation in the future. An example of a "rule" related to alethic necessity is: "It is necessarily true that each person can be born on only one date". Regardless of the denotation of the date in various different calendars, we are reflecting the reality that (in terms of everyday experience) a person can only be born once. Compare this with something like "It is necessarily true that the CEO of each company receives the highest remuneration of any employee in that company". Even if this is universally true for all companies that exist today, it may not have been so in the past, and it may not be so in the future, and so it is not a necessity. The rule (minus the erroneous "necessarily") would actually be contingent, not necessary. Adding words like "necessary" to contingent rules does not turn them into necessities – it just makes them badly formed.

Two common types of necessity are *physical necessity* and *logical necessity*. The first of these deals with situations governed by laws of nature in the real world (such as a person having one birth date). The second relates to situations where something is inevitably true by definition. For example, in an all-male society, it is necessary that the gender of every member of the society is male.

As mentioned, in SBVR necessity and impossibility are treated in the same way (since … *impossible that X* can be transformed into … *necessary that not X*). A necessity is defined in SBVR as a Structural Rule (synonym Definitional Rule). Possibility is classified in two ways, depending on whether the possibility is stated conditionally or unconditionally. A possibility that is stated conditionally (using the wording "… only if …") is called a Restricted Possibility Statement in SBVR and is classified as a Structural Rule, i.e., it is equivalent to a necessity. Unconditional possibility, non-necessity and contingency are not treated as rules. In SBVR, statements of these kinds these are deemed to be of a purely advisory nature (Statement of Advice of Possibility), intended only to provide guidance to the business. The fundamental difference between a rule and an advice is that "a rule always tends to remove some degree of freedom … contrasted with … 'advice', where a degree of freedom is never removed, even potentially".

The definitions in SBVR contain several puzzling features, which we mention briefly here. Many of the examples of necessity given in SBVR do not actually seem to be necessities. For example "It is necessary that each rental has exactly one requested car group" (in the context of a car rental company) may be a rule in a particular rental company, but there would seem to be nothing that would prevent another company allowing requests for multiple car groups, or the same company from changing its mind about car groups at some point in the future. In other words, the statement is not true in all possible worlds and hence not a necessity but a contingency.

However, contingency is treated in SBVR simply as a way of making rather vacuous statements such as "It is possible but not necessary that a renter's age is less than 21 years." This implies that we cannot use contingent statements to derive a particular truth-value. For example, in the above case, if we know that a person is a renter we can infer nothing about their age. However this rather misses the point that contingent statements can be true or false. For example "London is the capital city of England" happens to be true at the present time, and therefore this is not impossible. It is also

not necessary because it is not true in all possible worlds (the capital was earlier Winchester).

2.3 Deontic Modality

Deontic logic was developed to provide a formal basis for discussing concepts such as whether a particular situation is or is not obligatory, permitted, prohibited, and so on. It is a comparatively recent invention, originating only in the middle of the 20th century. By analogy with the alethic scheme outlined above, we can identify five deontic statuses.

- Obligatory: It is obligatory that X
- Permitted: It is permitted that X (it is not obligatory that not X)
- Optional: It is not obligatory that X and not obligatory that not X
- Prohibited (Impermissible): It is prohibited that X (it is obligatory that not X)
- Non-obligatory (Omissible): It is not obligatory that X

Similarly to its treatment of alethic modality, SBVR does not include explicit concepts for the Prohibited and Non-obligatory statuses, but uses negation to fold these back into the Obligatory and Permissible statuses. Similarly to the alethic case, it does include Prohibition and Non-obligation statements.

The parallel with the alethic modality is seductive, but needs to be treated with caution. For example, in the alethic case we typically assume "if X is necessary then X". However we could not extend this to the deontic case to assume "if X is obligatory then X".

The categorization scheme in SBVR for deontic modality closely parallels the scheme for alethic modality. As mentioned, Obligation and Prohibition are treated in the same way (since … *obligatory that X* can be transformed into … *prohibited that not X*). An obligation is defined in SBVR as an Operative Business Rule (synonym Behavioral Business Rule). Permissibility is classified in two ways, depending on whether it is stated conditionally or unconditionally. A permission that is stated conditionally (using the wording "… only if …") is called a Restricted Permission Statement and is classified as an Operative Business Rule, i.e., it is equivalent to an obligation. Unconditional permission, non-obligation and optionality are not treated as rules. Statements of these kinds are deemed to be of a purely advisory nature (Statement of Advice of Permission), intended only to provide guidance to the business. For reasons of space we will not discuss issues raised by this scheme since it closely parallels the alethic case.

2.4 The Use of Modalities in SBVR

We have seen that SBVR defines two main categories of rules: Structural Rules and Operative Rules, aligned respectively with alethic necessity and deontic obligation. Some of the non-normative sections of the standard hint that the rationale underlying this categorization is that a business would always comply with Structural Rules but may or may not comply with Operative Rules.

If we consider the business applicability of Structural Rules in the context of an information system we encounter a dilemma. Outside of an information system, no

business states necessities. There is no point in doing so, because the necessity will always be the case, regardless of whether the business does or does not makes any statement about it. Even a business statement countermanding the necessity will have no effect. For example, a proposed rule such as "each employee must be born on more than one date" would be ineffective because employees would still continue to have one birth date regardless. So we conclude that Structural Rules will not be found in the world outside an information system and must instead refer to the necessities found in information systems. But as noted above, nothing is necessary inside a business information system; such systems can contain whatever their design permits. So Structural Rules cannot be about the content of information systems. Thus, whether we look inside or outside of information systems, we find that the business expresses no rules that are necessities: therefore, we find no Structural Rules, at least not as SBVR defines that rule type. We later consider a solution to this dilemma, outside the framework of SBVR.

We next consider how some typical business rules might be classified in the SBVR scheme. The car rental company used as an example in the SBVR specification appears to have no specific limitation on the age of a renter, but we can imagine another car rental company that chooses to introduce a rule such as "The age of each renter must be 21 years or greater". This is not expressed as obligation, and so does not meet SBVR's criteria for an Operative Rule. It is clearly neither a necessity nor impossibility, because it is possible but not true in all possible worlds, and so it cannot be one of SBVR's Structural Rules. The only possible alethic status for this rule is contingent. In SBVR this must be categorized as an Advice, but an Advice cannot express a constraint (which this rule obviously does). The end result is that SBVR appears to provide no classification under which such a rule could fit. This is a considerable omission, since most businesses are likely to have many rules of this kind.

We could potentially bring the rule within the scope of SBVR by slightly modifying how we express it. For example, the rule could be treated as a Structural Rule if expressed in a form such as "A person can be considered a renter only if the age of the person is 21 years or greater". Alternatively, the same rule could be treated as an Operative Rule if expressed in a form such as "It is obligatory that the age of each renter must be 21 years or greater". However, such circuitous tricks should not be necessary, and in any case we might be nervous about a scheme that flips the same constraint between different categories as we try to re-word it.

But many of the "necessities" described in SBVR are obviously incorrect. One example presented in the specification is:

Necessity: A Customer has at least one of the following:

- a Rental Reservation.
- an in-progress Rental.
- a Rental completed in the past 5 years.

There is absolutely nothing necessary about these conditions: A company could easily choose a completely different definition of Customer. Even if a company adopted exactly this definition it would be free to change it at any point. Since Structural Rules are equated with necessities, the persistent misuse of necessity raises serious doubts about the value of Structural Rule as a category.

3 Rule Classification in Practical Information Systems

In this section we consider the impact of the way that rules are classified on the way that they are treated in practical information systems: typically (but not necessarily) systems that involve automation. We are not concerned here about specific automation components such as databases, rules engines, application programs, and so on, although we may use some of these to provide concrete examples. Our discussion simply assumes that any information system can be arranged to contain appropriate means to control or react to its own content.

3.1 The Practical Use of SBVR Rule Categories

We can illustrate how SBVR rule categories affect the potential realization of rules in an information system by means of an example. "Realization" here has its normal dictionary sense of the fulfillment or achievement of something desired by giving an actual form to a concept.

Consider the following two SBVR-style rules:

SR1: It is necessary that each employee has at most one birth date.
OR1: It is obligatory that each employee is assigned to at most one department.

We have introduced "… at most one…" to allow for situations where we do not know the birth date or department. The business may decide to base their design of a database partly on these rules, allocating a column for date of birth and a column for assigned department in a suitable table. In SQL [4], the columns can be defined to be single valued and NULLable (allowing the relevant table cells to be empty). The result is illustrated in Fig 1.

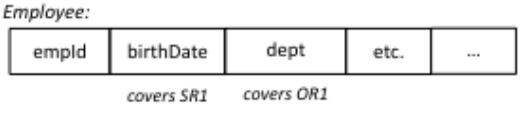


Fig. 1. Columns for Employee table

If we make ‘empId’ a key for the table (i.e., each employee has a unique 'empId' value) this design makes it impossible to enter more than one birth date or more than one department for any employee, covering the constraints expressed in SR1 and OR1. The important point to note here is that both SR1 and OR1 can be realized in exactly the same way. In other words, the distinction between Structural Rule and Operative Rule became irrelevant once we moved to their practical realization. In general, we find that the same range of possibilities for the practical realization of rules exists regardless of whether they fall into the SBVR categories of Structural Rule or Operative Rule, which suggests that this scheme may not be particularly useful when we come to build practical information systems.

3.2 The Deontic Nature of Information System Rules

Given the major status accorded to alethic Structural Rules in SBVR it may seem odd if alethic modality is not directly represented in a working information system. But consider the following argument.

1. (Here, some deontic rule—call it "Rule A"—about how X must be done, in this business. Examples: "Due to the pertinent alethic necessities, we should not record more than one birthdate, nor more than one gender, per student"; "Due to accreditation rules, we should not enroll more than 30 students in any given class-offering".)
2. (considered as a rule or perhaps just as a fact of the business) "In this business, we do 'X' by means of an automated, programmable system." (Examples: "We enroll students by means of our RDBMS"; "We keep track of students' birthdates and genders by means of (this same) RDBMS".)
3. "It is (therefore) very likely low cost, and fairly easy, to automate the observance of Rule A."
4. (a deontic rule): "If it's low-cost and fairly easy to automate the observance of a given rule, we should do so."
5. (a deontic rule implied by the above): "We (very likely, therefore) should automate the observance of Rule A--in the cheapest and easiest way."

Rules whose observance we should consider "automatable" (i.e. realizable), in this business, are those for which the final conclusion in (5) is reachable.

We can review this common, if often implicit, business logic against what seem to have been the original intentions of SBVR. "Rules" about which the business has no choice (because they are there anyway) are considered alethic necessities, and those that the business considers itself obliged to impose (because otherwise they would not be in effect) are considered deontic obligations. But from the point of view of the logic that the business follows, we can see that ALL business rules, even those arising from true necessities (unlike "Structural Rules"), are actually deontic: a business has choices about whether or not to include them in an information system, how they should be dealt with in the information system, and so on.

3.3 The Value of the Concept "Level of Enforcement"

SBVR includes the concept of a "level of enforcement" for Operative Rules, which it defines as "a position in a graded or ordered scale of values that specifies the severity of action imposed in order to put or keep an operative business rule in force". SBVR is unclear about whether a level of enforcement is optional or is required for every Operational Rule. The example set of levels given in SBVR differs from the definition in that some "levels" are actually differentiated by when actions are carried out, rather than severity. Structural Rules are defined to have no level of enforcement, but this is not a problem in practice, because in the context of an information system these rules are actually deontic. However, "level of enforcement" (as defined) does not seem to be particularly helpful when we consider the practical realization of rules in a business information system.

3.4 A Potentially More Useful, But Irregular, Use of the Two Modal-Logic Categories

Although from a modal-logic standpoint, all business rules are actually deontic, there is at least one standpoint from which we may perceive a practical value to categorizing rules according to the categories "alethic" vs. "deontic". But this categorization is irregular, as it does not pertain to the way the rule is intended (i.e., that which speech-act theory calls illocutionary force), but to the goal or purpose of the rule (i.e., that which speech-act theory calls perlocutionary force).

Specifically, we could understand the categories "alethic rule" and "deontic rule" to refer, not to the rule's modality, but to the purpose or goal of the rule: If the rule (e.g. a rule about how gender-records should be kept) is intended to keep the information system in line with some alethic reality (e.g. the fact that no one has two or more genders at a time), then we can categorize it as "alethic" (in quite an irregular sense of the term); otherwise, we can categorize it as "deontic" (in an equally irregular sense, since, strictly speaking, *all* business rules are deontic). The value of such a categorization scheme is that it does say something about the likely stability of the rule: since alethic realities can never change, a rule pursuant to the information system's conformity with such a reality, has a firmly stable motivation, and practically never should change; whereas other rules are much less stable, for they may change at the business's discretion. However, as we hope to show in a future paper, there is potential for an even more useful business-rule categorization scheme.

4 Conclusions

SBVR uses modal logic in its own idiosyncratic way. In particular, its treatment of necessity as something that happens to be true at a given time is very different from the generally accepted usage. But as discussed, and which is our main finding, this seems to reflect a more fundamental misunderstanding of how alethic modality relates to business rules.

A large proportion of SBVRs discussion of business rules is taken up by its categorization scheme, inspired by concepts taken from modal logic. As we have discussed in this paper, the SBVR scheme is confusing and appears to have little relevance to practical information systems. The OMG standard for Production Rule Representation [8] observes "… it is unclear if any of the conceptual distinctions of the SBVR business rule metamodel can be preserved in a PRR rule model". Information professionals may prefer to consider alternative categorization schemes if categorization is deemed to be important. For example rules could be categorized by which business function they impact (Accounting, Marketing, Production, etc.), by the business roles that have authority to update them, by business unit, and so on. And, of course, the same rule could exist in multiple categories.

However, a further range of classification can be based on the various ways in which rules can be realized in a practical information system. In a future paper we intend to explore, along these lines, the possibility of a more useful business-rules categorization scheme.

References

1. von Halle, B.: Business Rules Applied: Building Better Systems Using the Business Rules Approach. Wiley, Chichester (2001)
2. Morgan, T.: Business Rules and Information Systems. Addison-Wesley, Reading (2002)
3. Ross, R.: Principles of the Business Rule Approach. Addison-Wesley, Reading (2003)
4. SQL 2008: ISO/IEC 9075(1-4,9-11,13,14) (2008)
5. RuleML, http://ruleml.org/ (accessed February 2011)
6. Rule Interchange Format (RIF), http://www.w3.org/2005/rules/wiki/RIF_Working_Group (accessed February 2011)
7. SBVR, http://www.omg.org/spec/SBVR/1.0/ (2008)
8. PRR (2009), http://www.omg.org/spec/PRR/1.0/
9. Business Rules Group: Defining Business Rules ~ What Are They Really?, http://www.businessrulesgroup.org/first_paper/br01c0.htm
10. Stanford Encyclopedia of Philosophy, http://plato.stanford.edu/ (accessed February 2011)

Towards Modeling Data Variability in Software Product Lines

Lamia Abo Zaid and Olga De Troyer

WISE Lab, Computer Science Department
Vrije Universiteit Brussel (VUB)
Pleinlaan 2, 1050 Brussel
Belgium
{Lamia.Abo.Zaid,Olga.DeTroyer}@vub.ac.be
http://wise.vub.ac.be/

Abstract. In this paper, we provide an approach for modeling data variability as part of the overall software product line modeling approach. Modeling data variability in software product lines allows tailoring the data to the variability of a product. For this purpose, we have extended our Feature Assembly Modeling technique with the concept of *persistency feature*. We explain how these persistency features can be used to express the data variability, how they can be created and how they relate to the other features of the software product line. We also show how to derive a so-called *variable data model* from these persistency features and how an actual data model for a product of the product line can be derived. Additionally, annotations provide traceability between the variability of the features and the variability in the data model.

Keywords: Data intensive SPLs, Variable Data Model, Database Variability.

1 Introduction

Software product lines [1] allow organizations to deliver different variants of a product to different customers in a structured way. Software Product lines are based on the idea of creating different products from a set of shared features. These features can be composed differently to create different products. Some features are common to all products, while others exist only in some of the products (i.e. if the functionality they provide is required by that product).

Software product lines have found their way in many application domains, for example: eHealth systems [2], Mobile phones [3], Revenue Acquisition Management solutions [4], and Web portals [5]. In all these examples, the product line deals with data one way or another: processing data, generating and storing data, or simply retrieving data. To differentiate software product lines that deal with a large amount of persistent data and those that don't, we will call the former *data intensive product lines.* Although much attention has been given to specifying software features and their variability, very little attention has been given to how the variability in product lines affect the variability of the data.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 453–467, 2011.

Development of efficient data intensive software product lines requires an alignment between the features of an individual product and the data on which these features operate. Different features may require different parts of the data. For different reasons, such as reducing the complexity of queries, increasing performance, or ensuring security, data that is not needed by the features of a certain product could be labeled to be discarded in some way or the other in the final product. For instance, different features selected for a certain product may require different views on a global database or different customized databases may be required for different products. This motivates the need to provide variability specifications at the data level and to have a link between the features of a product line and the data associated with it. In this paper, we will focus on data-intensive software product lines that maintain their data in a database. We call a data model (or schema) that supports variability a *variable data model (or schema)*. In this paper we propose an approach for modeling data variability as part of the overall modeling of the software product line variability. The proposed approach is based on extending our Feature Assembly Modeling technique [6] with so-called *persistency features*. Persistency features are features that represent persistent data within the application domain. We also show how the corresponding variable data model can be defined from these persistency features.

This paper is organized as follows: in section 2, we provide a brief overview of related work. In section 3, we present our approach for modeling data variability in software product lines. We demonstrate the approach on an example. Next, in section 4, we show how this information can be translated to the corresponding variable data model. In section 5 we show how this variable data model can be used to derive databases tailored to the needs of the different products of the software product line. Finally, section 6 provides the conclusion and future work.

2 Related Work

Tailoring the database to the specific needs of a database actor is a well-known issue in database design. Often many views are created to suit the specific needs of different users or user groups (i.e. actors). For example, in [7] *schema tailoring* was proposed to meet the needs of different actors accessing different portions of data, in different usage scenarios. Data views were tailored for different actors in different contexts. It amounts to cutting out the appropriate data portion that fits each possible actor-context.

In [8] a component based technique for creating Real Time Database Management System (RTDBMS) was proposed to help resolve the complexity of creating different types of automotive control systems. An aspect-oriented methodology was adopted to relate the components to the functionality provided by them. The proposed platform consists of a library of components and aspects, and is supported by a tool suite. The tool suit assists system designers in configuring and analyzing different configurations based on the specific requirements of the targeted automotive system.

The issue of matching the database with each member of the product line was first raised in embedded systems [9] [10] [11], where the hardware is diverse and only limited resources exist. Therefore it is very important that the application and its accompanying database, as well as the database management system are suitably

tailored to meet the different requirements. In [9] the tailoring process was done at runtime to provide a configurable real-time database platform (database and DBMS). In [10], a product line approach was adopted to develop both the application and the suitable database management system (and also data) for each product. In that case, it was crucial that with each product of the product line only the essential data management requirements and essential data existed. The focus was given to the variability of the DBMS features. The authors did not mention how the database entities were affected by this variability in DBMS features. In [11], a feature oriented programming approach for tailoring data management for embedded systems was proposed in which a feature model describing the DBMS features and their variability was created. Feature oriented programming was used to create a common architecture and code base that allowed to configure different configurations of the DBMS (the approach was applied to Berkeley DB).

In [12], the authors reported the need for a variable database schema to serve the different needs of the product line. Neglecting this need leads to a gap between the application and the database. The authors proposed to tailor database schemas according to user requirements. Two methods were proposed, *physically decomposed schemas* (i.e. physical views) and *virtual decomposed schemas* (i.e. virtual views) for representing variability in the application and matching this variability with variability in the corresponding database. Once a product is configured (i.e. the features of the product are identified) the schema is tailored to meet the needs of the product features. The proposed technique decomposes an existing database schema in terms of features. It allows tracing of the schema elements to the program features at the code level using a technique similar to the #ifdef statements of the C preprocessor.

In this paper we present an approach that allows modeling the variability of the data according to the variability of the application. Variability of the application is analyzed with respect to its influence on the corresponding data. Next, a variable data model is created which takes into consideration the need for tailoring the database based on the member features of each product at an early stage (i.e. data modeling stage).

3 Extending the Feature Assembly Modeling Technique to Model Data Variability

Feature oriented domain analysis [13] is used to analyze and model the capabilities of a software product line in terms of *features.* A feature represents a distinctive logical unit that represents a functionality, capability, or characteristic of the software. The product line is represented by a *feature model*. A feature model specifies features, their relationships, their dependencies, and how they relate to the variability of the product line. In the last two decades several feature modeling methods were defined for example: Feature-Oriented Reuse Method (FORM) [14], FeatureRSEB [15], Product Line Use case modeling for Systems and Software engineering (PLUSS) [16], Cardinality Based Feature Modeling (CBFM) [17], and Feature Assembly Modeling (FAM) [6].

In this paper we extend Feature Assembly Modeling (FAM) technique [6] to model variability for data intensive software product lines. Feature Assembly Modeling is a multi-perspective feature-oriented modeling approach. The multi-perspective approach adopted allows specifying software product lines from different perspectives. Each perspective considers one point of view to describe the variability of the software product line (e.g., the *System* perspective describes the variability from a general system point of view, the *Functional* perspective describes the variability from a functional point of view, the Graphical *User Interface* perspective describes the variability from the viewpoint of the Graphical User Interface). The purpose of using perspectives is to simplify the design process by adopting the principle of "*separation of concerns*". Each perspective only concentrates of the features and the variability relevant for that perspective. Note that FAM provides a variable and extensible set of perspectives. If a (pre-defined) perspective is not relevant for a given software product line it should not be used. Also new perspectives can be defined if necessary for a given software product line.

To allow modeling data intensive product lines, we have introduced a special perspective, called the *Persistency Perspective*. The persistency perspective focuses on describing the variability from the viewpoint of the persistent data. The features in this perspective, called *persistency features*, are defined from the point of view of their need for manipulating (creating, updating, deleting, querying) persistent data required by the different members of the product line. In principle, the feature models for the different perspectives considered are defined (modeled) independently. However, the feature model for the persistency perspective is derived from the feature models of the other perspectives, i.e. the persistency features are derived by inspecting the features in the other non-persistency perspectives modeling the product line. If a feature needs persistent data, a corresponding persistency feature is defined. Next, the persistency perspective model is completed by adding dependencies and relations between features and the consistency and completeness of the overall model should be validated. We will discuss each step in more details in the next sections. We will use a running example: a Quiz Product Line (QPL) application [6], which is variable software for making Quizzes and designed to meet the needs of multiple customers and markets. A Quiz application is mandatory composed of a set of system features namely: *Question*, *Layout*, *License*, *Report* Generator, *Operation Mode* and *Question Editor*. When applying the Feature Assembly Modeling methodology to QPL, the following perspectives were defined: a system perspective, a functional perspective, a user perspective and a graphical user interface perspective[1]. In this paper we focus on the persistency perspective.

3.1 Defining the Persistency Perspective

As indicated, the features in the persistency perspective are derived from features defined in the various other perspectives. For the QPL example these perspectives are: the system perspective, the functional perspective and the graphical user interface perspective. Each perspective contains features that might or might not be associated with persistency needs. If a feature has needs in terms of persistent data, a corresponding persistency feature should be created in the persistency perspective (if it

[1] For more information on these perspectives please refer to [6] [18].

does not already exist). So, this implies that the persistency perspective should be created after the creation of the other perspectives, or while creating these other perspectives. If a new perspective is added later, the derivation of persistency features for this new perspective should also be performed. The following steps describe the process of deriving persistency features:

1. **Select a perspective:** The starting point is the *system perspective* (bird's eye view of the product line), as it provides an overview of the key features composing the product line and which are often associated with persistent information.
2. **Inspect the perspective for features that represent or require persistent data:** For each such feature, define a corresponding persistency feature.

   - For example in the QPL's system perspective some features are directly concerned with persistent data such as the *Question* feature that refers to the set of possible question types. Therefore a corresponding *Question* persistency feature will be defined in the persistency perspective.
     Also the *Report Generator* feature manipulates persistent data. The *Report Generator* requires the information about a certain *quiz* taken by a certain *user* to generate a *report*. In this case, we have defined two persistency features to represent this need, i.e. the *Quiz* feature (which represents the information about a specific quiz) and the *User-Quiz Info* (which represents the information about a certain quiz taken by a specific user). Additionally, this analysis also indicates the need for a persistency feature called *User* that represents the information about the user taking a quiz.

3. **Define composition relations relating persistency features.** These composition relations represent whole-part relations between persistency features. They show how features relate to each other from a compositional point of view. Furthermore, these composition relations also allow expressing variability, as a composition relation can be *mandatory* or *optional.*

   - For example in the QPL example, the persistency feature *Quiz* is mandatory composed of the following features: *Question* and *Quiz Element Options* which define respectively the set of possible questions for a quiz and the possible details about the quiz (such as passing score, passing feedback, etc.). Additionally, this *Quiz* feature is optionally composed of a *Question Media* feature that identifies the set of possible media associated with a question. See figure 1(a) for the graphical representation.

4. **Introduce or distinguish between *abstract* and *concrete* persistency features.** Sometimes it may be useful to introduce an *abstract feature* [6] to generalize from a number of more specific features. In that case, the more specific features are called *option features* [6]; they are linked to the abstract feature using a generalization/specification relation. Abstract features must be associated with cardinality rules that govern the maximum and minimum number of option features that should be selected in a valid product configuration.

   - For example in the QPL example, the persistency feature *Question* is defined as an abstract feature that generalizes the features representing the different types of questions. Therefore, the *Question* feature has the following option features: *Sequencing Question*, *True/false Question*, *Matching Question*,

Multiple Choice Question, and *Fill the Blank Question*. We can specify that at least one question type has to be selected and there is no maximum limit in the number of question types used. So the cardinality is expressed as "*1:-*" (see figure 1(a)).

5. **Define the inter-perspective dependencies for the persistency perspective features defined so far.** Inter-dependencies express dependencies between features belonging to one single perspective.
   - For example within the QPL persistency perspective, if we consider the *Question* feature (figure 1.a) and the *Quiz* feature (figure 1.b), we want to express the constraint that for a *Question* feature, selecting the *Assessment Media* feature also requires selecting the *Question Media* feature. This translates into the following inter-perspective dependency: `Assessment Media requires Question Media.`
6. **Define the intra-perspective dependencies**, i.e. dependencies between the features of the perspective selected in step 1 and the features of the persistency perspective.
   - For example in the QPL example, selecting the *Self Assessment* feature (from the system perspective) implies that the data for the questions assessment should also be selected, i.e. the *Question Assessment* persistency feature should also be selected. This translates into the following intra-perspective dependency: `System. Self Assessment requires Persistent. Question Assessment`
7. Repeat steps 1 to 6 to extract persistency features from all existing perspectives (Functional perspective, User's perspective, etc.).

In this way, the persistency perspective is incrementally created.

3.2 Validating the Consistency and Completeness of the Overall Model

Due to the often-tangled relation between data and functionality, a validation for the persistency perspective generated in the previous step is recommended. This is a two-step process motivated by the work in [19] [20], as follows:

1. Validate the consistency of the persistency perspective and its associated inter-dependencies and refine when necessary. This means verifying the following:
   a. Check if no persistency features are unintentionally missing, i.e. the defined persistency features provide an overview of the data concepts required by features of the product line (in all perspectives). This missing of features may be due to errors made during the definition of the persistency perspective or due to missing features in the other perspectives. In this way, this validation is also to some extent a validation of the completeness of the other perspectives.
   a. Check for duplication of persistency features or redundant persistency features, i.e. whether some features have the same semantics (based on their decomposition hierarchy) and actually represent the same persistency feature (this can occur because the same persistency features may originate from different perspectives). In case there is a need for this duplication the "*same*" [21] dependency should be used to treat these redundant features as one.

b. Validate the consistency of the defined inter-dependencies between features of the persistency perspective by checking whether conflicts exist among them (taking into account features related by "same" dependency) yielding to conflicts within the one perspective.

2. Validate the intra-dependencies between the persistency perspective and the other perspectives. This is to some extent a validation of the overall model, it means verifying the following:
   a. The intra-dependencies between features of the different perspectives and the persistency perspective are complete. For this purpose the overall global model needs to be inspected. The global model is created by using the "*same"* dependency as a merge operator that enables merging perspectives based on their common features. This merging of common features allows linking together the different perspectives based on their feature commonality. Furthermore the intra perspective dependencies link features of one perspective to their related features in other perspectives. This linking of related features as well as merge of common features allows gluing together the different perspectives and therefore obtaining a global model.
   b. Within the global model identify and resolve any conflicts found within the intra-perspective dependencies defined.

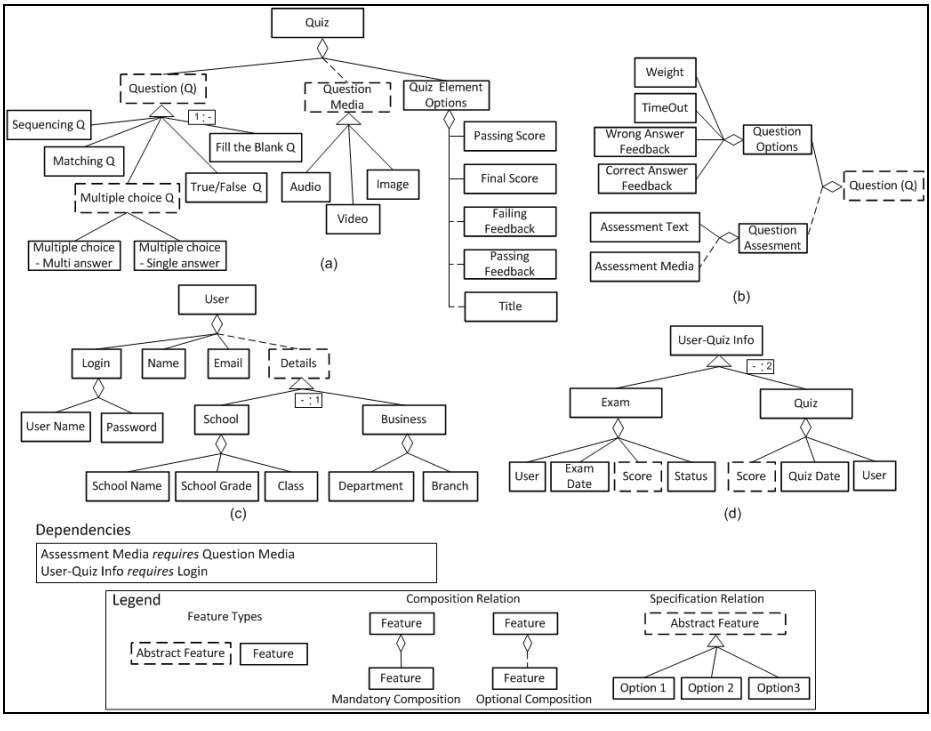


Fig. 1. Persistency perspective for the QPL

Figure 1[2] shows the result of applying this process to the Quiz product line, Figure 1.a shows the *Quiz* persistency feature, Figure 1.b shows the *Question* persistency feature, Figure 1.c shows the *User* persistency feature (this was mainly derived from features in the User perspective and the Functional perspective), and Figure 1.d shows the *User-Quiz Info* persistency feature which represents information about a specific quiz taken by a specific user. Figure 1 also shows the (inter-perspective) dependencies between the features of this persistency perspective.

- `System.Exam requires Persistent.Termination Page`
- `System.Muliuser requires Persistent.User`
- `Users.School requires Persistent.School`
- `Users.Bussiness requires Persistent.Bussiness`
- `System.Self Assessment requires Persistent.Question Assessment`

Listing 1. Intra-perspective feature to feature dependencies for the persistency perspective.

As already mentioned, it is also important to identify how features affect each other, in terms of intra-perspective feature-to-feature dependencies which tie together the different (related) perspectives to create a global model of the system. The set of intra-perspective feature-to-feature dependencies for the persistency perspective of the quiz product line is given in Listing 1.

Once the persistency perspective is defined and all perspectives are consistent, the next step is to use this information in the database modeling process to obtain a *variable data model*, which can serve for tailoring the different databases required for the different possible product line members (i.e. products). We describe this in the next section.

4 Establishing a Variable Data Model

The persistency perspective is used to steer the database modeling process in producing a variable data model, i.e. a data model that reflects the different variability needs of the product line.

In general, two scenarios exist for defining a data model: a *centralized design* and a *decentralized design* [22]. In centralized database design, the database model is defined in one step, and as a result one global database model is defined. In decentralized database design, a data model is defined for each user view resulting in a number of data model views. In case a global data model is required, it can be derived via a view integration process where the different segments of the database design are combined to create one global model.

How the database model is defined does not affect how the link between the variability model (which is represented by the Feature Assembly Models) and the data

[2] The legend shown in figure 1 is applicable for all subsequent figures showing Feature Assembly Models.

model is achieved. In either case, variability information is associated with the data model to instruct how to derive the possible different data models for the different products of the product line. The data model incorporating variability information is called the *variable data model*. There are basically two options for tailoring a variable data model to the needs of each product: the *materialized view* and the *virtual view* [23]. A materialized view means that the required data model elements are actually extracted from the variable data model and are stored physically. This allows creating a tailored database for the final product composed of tables materialized by these views. While with the virtual view, a central database exists, and each individual product has its own view (or sets of views) on this common database.

Modeling a variable data model is a two-step process; first *persistency features* are mapped into *data concepts*. Secondly, the variability of these data concepts is explicitly specified in the defined data model.

4.1 Mapping Persistency Features to Data Concepts

The persistency perspective provides the information about the required persistent data and their variability. Therefore, features in the persistency perspective are mapped to data concepts. To illustrate the mapping, we will use the EER model [24] to represent the data model. However, any other data modeling techniques (such as ORM [25] or UML [26]) can also be used.

All features in the persistency perspective should map to a corresponding data concept. As we are using EER, a data concept can be either an *entity* or an *attribute*. We have defined two annotation to represent this mapping, namely `<<maps_to>>` and `<<relates_to>>`.

`<< maps_to>>` establishes a one-to-one mapping between a feature and a data concept (i.e. entity or attribute).
For example:

- `Persistent.Question <<maps_to>> Data_Model.Question`
- `Persistent.Passing_Score <<maps_to>>`
  `Data_Model.Quiz.Passing_Score`

It should be noted that data concepts could be *related* to any feature within any of the existing perspectives. To describe this kind of relationship we use the `<<relates_to>>` notation. `<< relates_to>>` identifies an *association relation* between a feature and a data model concept (i.e. entity or attribute). These association relations are important to establish a link between features in general and the data model.
For example:

- `Functional.Question Category <<relates_to>> Data_Model.Category`

Basically, persistency features map to either *entities* or *attributes* in the data model. Guidelines are:

a. Key features (i.e. features that represent a concrete concept or object) map to entities. For example *Persistent.Question* maps to a *Question* entity in the data model.

b. Features expressing details of key features are in general mapped to attributes rather than entities. For example *Persistent.Passing_Score* maps to the *Passing_Score* attribute of the *Quiz* entity.

We now explain how we can indicate the variability of the data model.

4.2 Representing Variability in Data Models

To specify variability of a data model, it was necessary to extend the model notation. For the EER model, this has been done as follows: we introduced annotations that are used to mark the variability of the concepts, i.e. *entities*, *attributes*, and *relations*. A data concept that originated form a *variable* persistency feature (i.e. optional feature) should be a variable concept in the database model. To denote variability of an entity or attribute, it is annotated with `<<variable>>` and we call it a *variable concept*, i.e. its existence is dependent on the selection of the corresponding feature in the product line. For example, a *Question* feature can be optionally composed of *Question Options* (e.g., time out, weight …etc.). In this case, the *Question* entity is linked to an entity named *Question Options* that is marked as variable, i.e. *Question Options* is marked with the `<<variable>>` annotation to indicate this variability.

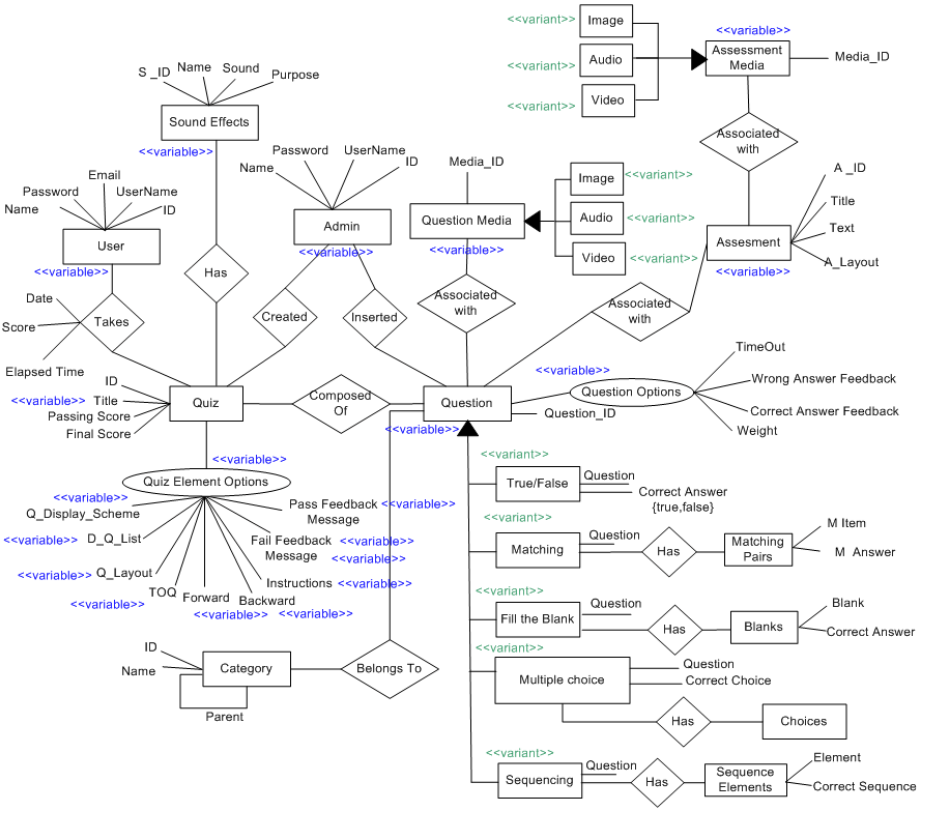


Fig. 2. The variable EER data model for the Quiz product line

A data concept that originates from an *option* persistency feature should also be a variable concept in the data model. To denote variability that is due to a specialization relationship (i.e. option feature), the annotation <<variant>> is used. For example in the QPL persistent perspective, there are five different option features of *Question*: *Sequencing, True-False, Matching, Fill the Blank,* and *Multiple Choice*. These *Question* option features are mapped to entities (each feature is represented by an entity) and each entity is annotated with <<variant>> to indicate their variability and the fact that they are derived from option features (see figure 2).

Please note that the resulting data model may not be complete. Features defined in the persistency perspective may not completely define all entities, attributes, relationships and constraints of the data model. Rather they only define those concepts related to variability of the product line. Therefore, it is up to the data designer to complete the data model.

Figure 2 shows the variable EER data model for the QPL example. Figure 2 shows that attributes as well as entities can be annotated as variable database concepts. Note that when an attribute is marked as *variable* it should not be used as a *primary key* because it is not guaranteed to exist in all variants of the schema. For the same reason, when using it as a *foreign key* care should be taken as in some schema variants it may not be used. (Note that if no non-variable key is available an artificial key can be introduced.) Also it may be useful to consider putting variable attributes in a separate entity than the primary entity they are attributes of. This will allow easier tailoring of the data model when the variable attributes are not selected. *Quiz Preferences* is an example of such practice.

5 Deriving Tailored Product Data Schemas

As we have explained, a *variable data model* defines the variability of the concepts involved in the variability of a software product line. However, it does not only define the variability of the concepts, it also links the variability of these concepts to the variability of the application features. This enables traceability, i.e. it makes it possible to trace the variability from application to database and vice versa. This allows tailoring the data schema of each product to match the data requirements of its features.

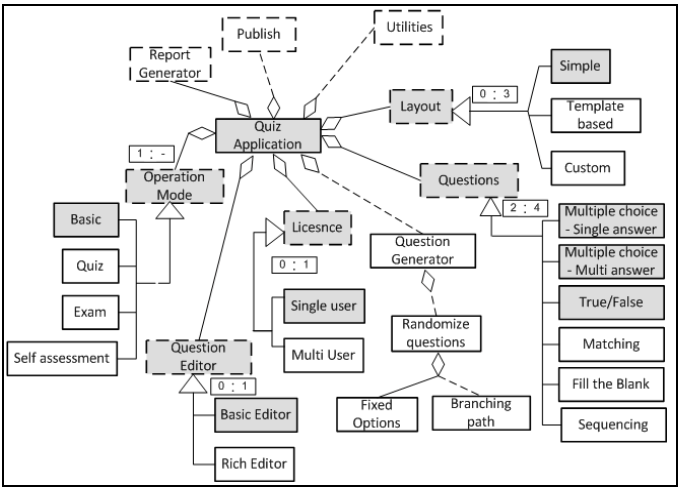


Fig. 3. System perspective of Quiz product 1

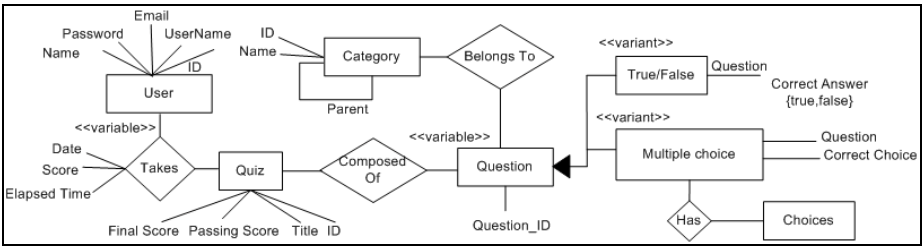


Fig. 4. EER data model for product 1

To demonstrate this, we defined two possible products of the quiz product line, Quiz product 1 is a simple quiz, that contains the following system features *[Quiz, License {Single User}, Layout {Simple}, Questions {True/False, Multiple Choice}, Operation Mode {Basic}, and Question Editor {Basic Editor}],* figure 3 shows in grey shade these selected features . Figure 4 shows the EER data model (view) for Quiz product 1. In listing 2, the relevant mappings are shown, which express the link between the data model and the features.

```
System.Quiz Application <<relates_to>> Data_Model.User
System.Quiz Application <<relates_to>> Data_Model.Quiz
Persistent.Questions  <<maps_to>> Data_Model.Questions
Persistent.True/False  <<maps_to>> Data_Model.True/False
Persistent.Multiple Choice Single Answer <<relates_to>> Data_Model.Multiple Choice
Persistent.Multiple Choice Multi Answer <<relates_to>> Data_Model.Multiple Choice
```

Listing 2. Feature Assembly-to-data model mappings of Quiz product 1

Quiz product 2 shows a more complex example. It is a Quiz and Exam application, and it contains the following system features *[Quiz, License {Multi User}, Layout {Simple, layout, Template Based}, Questions {Sequencing, Matching, Multiple Choice}, Operation Mode {Quiz, Exam}, and Question Editor {Rich text editor}, Report Generator]*. Figure 5 shows the selected features (in grey shade) that represent Quiz

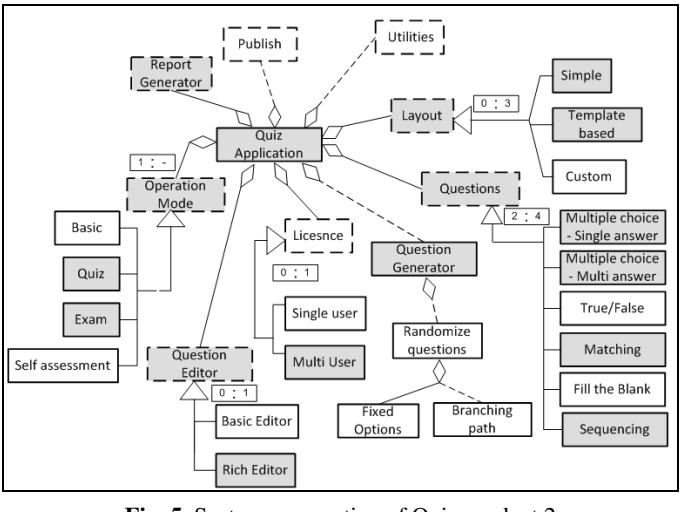
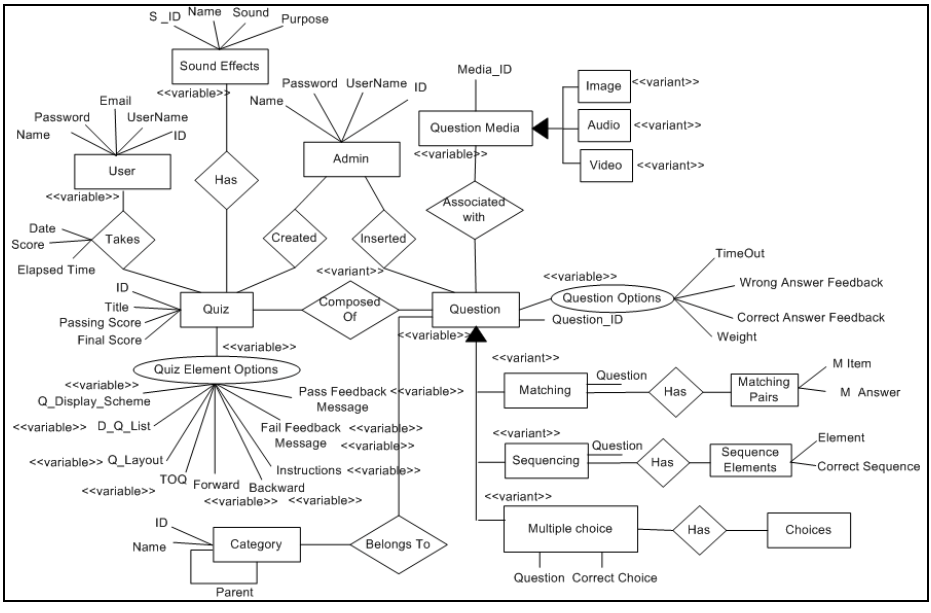


Fig. 5. System perspective of Quiz product 2

product 2. Figure 6 shows the EER data model (view) for Quiz product 2, and listing 3 contains the relevant mappings.



EER data model for product 2

```
System.Quiz Application <<relates_to>> Data_Model.User
System.Quiz Application <<relates_to>> Data_Model.Quiz
Persistent.Questions  <<maps_to>> Data_Model.Questions
Persistent.Multiple Choice Multi Answer <<relates_to>> Data_Model.Multiple Choice
Persistent.Matching <<maps_to>> Data_Model.Matching
Persistent.Sequencing <<maps_to>> Data_Model.Sequencing
System.Multi User <<relates_to>> Data_Model.Admin
System.Multi User <<relates_to>> Data_Model.Question Options
System.Multi User <<relates_to>> Data_Model.Quiz Element Options
Persistent.User <<relates_to>> Data_Model.User
System.Exam <<relates_to>> Data_Model.Sound Effects
System.Multi User <<relates_to>> Data_Model.Question Media
```

Listing 1. Feature Assembly-to-data model mappings of Quiz product 2

3 Conclusion and Future Work

In this paper we have discussed the need for modeling data variability as well as application variability when designing software product lines. We have extended the Feature Assembly Modeling technique used for feature modeling to also support data variability. For this purpose, we have defined a *persistency perspective* in which fea-

tures are defined from the point of view of their need for manipulating persistent data in the product line. This persistency perspective is derived from the other perspectives defining the product line. We have also shown how the persistency perspective can be used to define a *variable data model*. A variable data model is a data model annotated with variability information. As such, the actual data model for a product of the product line can be tailored to meet only the requirements of this specific product. This may simplify the process of accessing data, resulting in simpler queries and avoids dummy values in insert and update queries. Another advantage is that it optimizes storage space and allows for different implementation (e.g., for a simple product a full fledged DBMS may not be needed) In addition, the link between the features of the product line and the variable data model is maintained, such that automatic derivation of the actual data model for an individual product is possible.

In this paper, we used and extended EER to express the variable data model, however in a similar way, other data modeling techniques (e.g., UML, ORM) can be used.

Future work includes tool support for the modeling the persistency perspective (this is part of the tool support for the overall feature assembly modeling approach) and for supporting the mapping to the variable data model. We also plan to apply the approach to an industrial case to validate it further.

Acknowledgement

This research is sponsored by IRSIB through the VariBru project (www.varibru.be).

References

1. Bosch, J.: Design and Use of Software Architectures: Adapting and Evolving a Product-Line Approach. Addison-Wesley, Boston (2000)
2. Bartholdt, J., Oberhauser, R., Rytina, A.: An Approach to Addressing Entity Model Variability within Software Product Lines. In: ICSEA 2008, pp. 465–471 (2008)
3. Bosch, J.: Software product families in nokia. In: Obbink, H., Pohl, K. (eds.) SPLC 2005. LNCS, vol. 3714, pp. 2–6. Springer, Heidelberg (2005)
4. Clements, P.C., Northrop, L.M.: A Software Product Line Case Study. Technical Report CMU/SEI-2002-TR-038 (November 2002)
5. Pettersson, U., Jarzabek, S.: Industrial Experience with Building a Web Portal Product Line using a Lightweight, Reactive Approach. In: ESEC-FSE 2005, European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering. ACM, New York (2005)
6. Abo Zaid, L., Kleinermann, F., De Troyer, O.: Feature assembly: A new feature modeling technique. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, pp. 233–246. Springer, Heidelberg (2010)
7. Bolchini, C., Quintarelli, E., Rossato, R.: Relational data tailoring through view composition. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 149–164. Springer, Heidelberg (2007)
8. Nyström, D., Tesanovic, A., Nolin, M., Norström, C., Hansson, J.: COMET: A Component-Based Real-Time Database for Automotive Systems. In: Proceedings of the Workshop on Software Engineering for Automotive Systems at 26th International Conference on Software engineering (ICSE 2004). IEEE Computer Society Press, Los Alamitos (2004)

9. Tesanovic, A., Sheng, K., Hansson, J.: Application-Tailored Database Systems: a Case of Aspects in an Embedded Database. In: Proceedings of the 8th International Database Engineering and Applications Symposium (IDEAS 2004). IEEE Computer Society Press, Los Alamitos (2004)
10. Rosenmüller, M., Siegmund, N., Schirmeier, H., Sincero, J., Apel, S., Leich, T., Spinczyk, O., Saake, G.: FAME-DBMS: Tailor-made Data Management Solutions for Embedded Systems. In: Proceedings of EDBT Workshop on Software Engineering for Tailor-made Data Management (SETMDM), pp. 1–6. ACM Press, New York (2008)
11. Rosenmüller, M., Apel, S., Leich, T., Saake, G.: Tailor-made data management for embedded systems: A case study on Berkeley DB. Data & Knowledge Engineering 68(12), 1493–1512 (2009)
12. Siegmund, N., Kästner, C., Rosenmüller, M., Heidenreich, F., Apel, S., Saake, G.: Bridging the Gap between Variability in Client Application and Database Schema. In: BTW 2009, pp. 297–306 (2009)
13. Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, A.: Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR-021, Software Engineering Institute, Carnegie-Mellon University (1990)
14. Kang, K., Kim, S., Lee, J., Kim, K., Shin, E., Huh, M.: FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. J. Annals of Software Engineering. 5, 143–168 (1998)
15. Griss, M., Favaroand, J., d'Alessandro, M.: Integrating Feature Modeling with the RSEB. In: Fifth International Conference on Software Reuse, pp. 76–85 (1998)
16. Eriksson, M., Börstler, J., Borg, K.: The PLUSS approach - domain modeling with features, use cases and use case realizations. In: Obbink, H., Pohl, K. (eds.) SPLC 2005. LNCS, vol. 3714, pp. 33–44. Springer, Heidelberg (2005)
17. Czarnecki, K., Kim, C.H.P.: Cardinality-Based Feature Modeling and Constraints: A Progress Report. In: OOPSLA 2005 International Workshop on Software Factories (2005)
18. Abo Zaid, L., Kleinermann, F., De Troyer, O.: Feature Assembly Modelling: A New Technique for Modelling Variable Software. In: Cordeiro, J., Virvou, M., Shishkov, B. (eds.) 5th International Conference on Software and Data Technologies Proceedings, vol. 1, pp. 29–35. SciTePress (2010)
19. Abo Zaid, L., Houben, G.-J., De Troyer, O., Kleinermann, F.: An OWL- Based Approach for Integration in Collaborative Feature Modelling. In: 4th Workshop on Semantic Web Enabled Software Engineering - SWESE 2008 (October 2008)
20. Sabetzadeh, M., Nejati, S., Liaskos, S., Easterbrook, S., Chechik, M.: Consistency Checking of Conceptual Models via Model Merging. In: 15th IEEE International Requirements Engineering Conference, RE 2007 (October 2007)
21. Abo Zaid, L., Kleinermann, F., De Troyer, O.: Applying Semantic Web Technology to Feature Modeling. In: The 24th Annual ACM Symposium on Applied Computing, The Semantic Web and Applications (SWA) Track (March 2009)
22. Connolly, T.M., Begg, C.E.: Database Systems: A Practical Approach to Design, Implementation and Management. Addison-Wesley, Reading (2009)
23. Gupta, A., Mumick, I.S.: Materialized views: techniques, implementations, and applications. MIT Press, Cambridge (1999) ISBN: 978-0262571227
24. Thalheim, B.: Extended Entity-Relationship Model. In: Encyclopedia of Database Systems 2009, pp. 1083–1091 (2009)
25. Morgan, T.: Information Modeling and Relational Databases, 2nd edn. Morgan Kaufmann Publishers, San Francisco (2008) ISBN: 978-0-12-373568-3
26. Fowler, M.: UML distilled: a brief guide to the standard object modeling language. Addison-Wesley, Reading (2004) ISBN-13: 978-0-321-19368-1

Experimenting with the Comprehension of Feature-Oriented and UML-Based Core Assets

Iris Reinhartz-Berger and Arava Tsoury

Department of Information Systems,
University of Haifa, Haifa 31905, Israel
iris@is.haifa.ac.il, aravabt@gmail.com

Abstract. Software product line engineering mainly deals with specifying and developing core assets that can be utilized and adapted into specific product artifacts. Feature-oriented and UML-based modeling methods have been proposed for managing and supporting core assets specification. While these methods get a lot of attention in software product line engineering literature, their comparison in terms of comprehension is somewhat neglected. Being suitable for early stages of core assets development, this work aims at performing comparative analysis and discussing their advantages and limitations in view of two main stakeholders: developers and product customers. To this end, we conducted two experiments for examining the comprehension of core assets specification in feature-oriented CBFM and UML-based ADOM. The results showed that the only significant difference in terms of comprehension between these methods is in variability specification; developers may better understand the locations at which variability occurs and the ways to realize variability in ADOM.

Keywords: variability, software product line engineering, domain analysis, UML, feature-orientation.

1 Introduction

Software product line engineering (SPLE) [21] deals with analyzing, designing, and implementing assets for families of software products, as well as customizing these assets to fit specific product requirements. For these purposes, ***core assets*** are defined as parts that are built to be used by more than one product in the family, while ***product artifacts*** are specific parts of the software products [2]. In order to be reusable and suitable to a wide variety of products, core assets have to specify both commonality and variability aspects of given software product families [25], [26].

Different methods have been suggested for specifying core assets in SPLE and other related fields. The main corpus of works includes feature-oriented and UML-based methods. Feature-orientation [16] supports specifying core assets as sets of characteristics relevant to some stakeholders and the relationships and dependencies among them. A product asset, representing an application or a software product, uses the reusable (core) asset and instantiates a sub-set of features that satisfies certain conditions. Variability is specified in terms of mandatory vs. optional features,

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 468–482, 2011.

alternatives, OR features, 'require' and 'exclude' dependencies among features, and composition rules.

UML-based methods extend UML metamodel or more commonly suggest profiles for handling core asset specification and variability modeling. They typically represent variation points, variants, and dependencies among them [13]. A *variation point*, which identifies a location in a core asset at which a variation may occur, can be associated to several *variants*, each of which realizes a possible way to create particular product artifacts. Variation points further provide guidance and validation aids for adapting core assets to particular contexts (product artifacts).

Although having different roots, both feature-oriented and UML-based approaches support specifying core assets at early development stages: feature-oriented methods enable modeling product line requirements mainly via feature diagrams, which are trees or graphs of potential characteristics and dependencies among them, while UML-based methods utilize different diagram types, most notably use case and class diagrams, for the same purpose. Thus, we wish to explore in this work the benefits and limitations of these approaches and to empirically compare their comprehension for various stakeholders in SPLE, namely developers and product customers.

To this end, we chose two specific methods: Application-based DOmain Modeling (ADOM), which is defined through a UML profile [22], and Cardinality-Based Feature Modeling (CBFM), which is a feature-oriented method [6]. We checked the comprehension of models specified using these methods with two populations of information systems students: novice students with no knowledge and experience in developing software products, who played the role of product customers, and advanced students who studies a domain engineering course and played the role of developers. In particular, we were interested in the following three research questions:

RQ1. The specifications of which method, CBFM or ADOM, are more comprehensible to developers?

RQ2. The specifications of which method, CBFM or ADOM, are more comprehensible to product customers?

RQ3. Are the specifications of CBFM or ADOM less comprehensible than text specifications to product customers?

We found out that although feature-orientation is considered simpler and thus more intuitive [14], [24], no significant difference was found in comprehending core assets in the selected methods, except of in comprehending the locations and realizations of variability: ADOM outperformed CBFM on this subject. The text specification was significantly better than CBFM and ADOM specifications only in commonality-related aspects of the given domain.

The remainder of this paper is organized as follows. Section 2 reviews existing feature-oriented and UML-based methods for modeling core assets and justifies the selection of CBFM and ADOM for this work. Section 3 elaborates on these particular methods and exemplifies their usage. Section 4 describes the empirical comparison, discussing the research questions, the settings, the results, and the threats to validity. Finally, Section 6 concludes and refers to future research directions.

2 Feature-Oriented and UML-Based Methods in SPLE

As noted, core assets need to capture both commonality and variability of product lines. Analyzing the SPLE literature, Table 1 summarizes the main aids that are required for specifying both commonality and variability aspects in software product lines. Additional aspects, such as binging time and reuse mechanisms, exist, but they are out of the scope of the current work as they are handled only in few methods. Note that we categorized optional elements and inter-dependencies as referring to commonality rather than to variability, since they aim at characterizing a large portion of the products in the family and do not provide explicit aids for managing variability. The rest of this section uses the concepts mentioned in Table 1 for comparing the methods in each category.

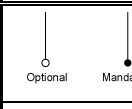
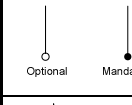
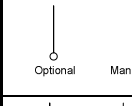
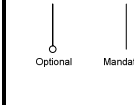
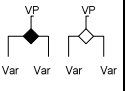
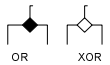
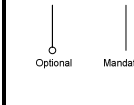
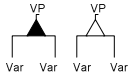
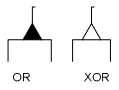
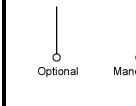
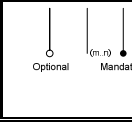
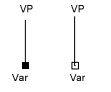
Table 1. Criteria for Comparing SPLE Methods

Category	Concept	Description
Commonality	Multiplicity: mandatory vs. optional elements	***Mandatory elements*** are elements that identify the product family. All products that belong to the family must include these elements. ***Optional elements*** may add some value to the product, when selected. However, not all the products that belong to the family will include them.
	(Inter-) Dependencies	(Inter-) Dependencies are restrictions for selecting groups of optional elements. Two commonly used dependencies are: (1) Requires: A → B and (2) Excludes (or mutual exclusive, mutex): A → ¬B.
Variability	Variability specification: variation points and variants	A ***variation point*** identifies a location at which a variable part may occur. It locates the insertion point for the variants and determines the characteristics (attributes) of the variability. ***Variants*** realize possible ways to create particular product artifacts at certain variation points.
	Selection rules (cardinalities)	***Cardinality*** specifies the minimal and the maximal numbers of variants that have to be selected in a given variation point.
	Addition rules (openness)	***Openness*** refers to the ability of a product artifact to add variants that were not specified in a variation point of the core asset. In particular: ***Open variation point*** allows specifying in the product artifact new variants which were not associated to the variation point in the core asset. ***Closed variation point*** do not allows such situations and all possible variants should be listed in the core asset and associated to the variation point.

2.1 Feature-Oriented Methods

As noted, feature-oriented methods primarily use feature diagrams for specifying core assets. The root of most feature-oriented methods is Feature-Oriented Domain Analysis (FODA) [16]. However, FODA has different shortcomings, the important of which are: (1) the semantics of the dependencies among features is unclear; (2) the specification of variation points and variants is not done explicitly; and (3) the selection of features is relatively limited and mainly refers to mandatory and optional elements. Thus, different extensions to FODA have been proposed to tackle these shortcomings. Table 2 summarizes the main extensions and compares them according to the expressiveness criteria listed in Table 1.

Table 2. Comparison of Feature-Oriented Methods

Method Name	Commonality		Variability		
	Multiplicity	Inter-Dependencies	Variability Specification	Selection rules	Addition rules
FORM [18] & FOPLE [15]		Textually, via composition rules. Graphically, via "implemented by" dependencies.	VP – NS Var – NS		NS
GP [7]			VP – NS Var – NS		NS
FORE [23]		"requires"	VP – NS Var – NS		NS
FeatuRSEB [11]		NS			NS
VBFD [26]		NS			NS
PLUSS [8]		"requires", "excludes"	VP – NS Var – NS		NS
CBFM [6]		Using OCL			NS

NS=Not Supported, VP = Variation Point, Var. = Variant

As can be seen, FORM, FOPLE, GP, FORE, and PLUSS do not explicitly specify variability via variation points and variants. However, selection rules are partially supported in these approaches via XOR and OR constructs. FeatuRSEB, VBFD, and CBFM support representing variation points and variants and refer to selection rules via OR and XOR constructs. All the reviewed methods do not support specifying addition rules, e.g., in the form of open and closed variation points. Nevertheless, the expressiveness of CBFM exceeds that of the compared methods in the inter-dependencies and selection rules categories: CBFM enables using OCL for specifying different inter-dependencies and it allows defining various cardinalities for specifying selection rules. Thus, we chose this method for our comparison and added to it the ability to specify addition rules, as will be described in Section 3.1.

2.2 UML-Based Methods

Most UML-based methods define profiles for specifying core assets, while some of them suggest small modifications to UML metamodel. Table 3 summarizes several UML-based methods and compares them according to the expressiveness criteria listed in Table 1.

Table 3. Comparison of UML-based Methods

Method Name	Commonality		Variability		
	Multiplicity	**Inter-Dependencies**	**Variability Specification**	**Selection Rules**	**Addition Rules**
PLUS [10]	Mand. – «kernal» Opt. – «optional»	NS	VP – NS Var – «variant»	By default, variants are mutually exclusive selected	NS
Halmans & Pohl [12]	Only for VPs: Mand. – filled triangle Opt. – unfilled triangle	NS	VP – triangle Var – «variant»	According to the triangle color and the relationship type (to variants)	NS
Ziadi et al. [30]	Mand. – default Opt. – «optional»	generic constraints	VP – «variation» (abstract classes) Var – «variant»	NS	NS
Alves de Oliveira et al. [1]	only for variants: Mand. – «mandatory» Opt. – «optional»	«requires», «mutux» (between variants)	VP – «variation point» Var – elements associated to VPs	«alternative_or», «alternative_XOR»	NS
Morisio et al. [20]	NS	NS	VP – «V» Var – inheritance to VP	The default is OR, «xorV» specifies mutual exclusiveness	NS
Robak et al. [24]	NS	NS	VP – NS Var – «variable» + tagged values	Specified via {or} and {xor} constraints on the associations	NS
SPLIT [5]	Only for VPs: The existence attribute indicates whether the VP is optional or mandatory	NS	VP – «variation point» Var – elements connected to VPs via associations	Through attributes of the VPs	Through attributes of the VPs
VPM [28]	Only for VPs: Mand. – m Opt. – o	NS	VP – vp V – «variant»	NS	Through callbacks and guidelines
Clauß [3], [4]	Mand. – by default Opt. – «optional»	«requires», «mutex»	VP – «variation point» Var – «variant»	Through the multiplicity tagged value	NS
ADOM [22]	Mand. – «mandatory» Opt. – «optional» «multiplicity min=m max=n»	«requires», «excludes»	VP – «variation point» Var – «variant»	Through the card tagged value	Through the open tagged value

NS=Not Supported, Mand.=Mandatory Elements, Opt.= Optional Elements,
VP = Variation Point, Var. = Variant

As can be seen, commonality is usually specified using dedicated stereotypes for differentiating mandatory (sometimes called kernel) and optional elements. Some works explicitly specify variability using both «variation point» and «variant» stereotypes, while others specify only one of these concepts and the other is implicitly

specified from its relationships with the other concept. Several works explicitly refer to dependencies between elements in the form of stereotypes, tagged values, or constraints.

We chose ADOM for our empirical study, since it explicitly refers to the selection and addition of variants in certain variation points, aspects which other UML-based methods tend to neglect. Furthermore, it enables explicit specification of both variation points and variants and it allows specifying ranges of multiplicity via the «multiplicity» stereotype.

3 The Compared Methods: CBFM and ADOM

This section is devoted to the introduction of the compared methods, CBFM and ADOM, and to their exemplification through a domain of virtual offices (VOF). Virtual offices are a type of telecommunication in which workers are occupied with the tools, technologies, and skills to perform their jobs from anywhere in the world, e.g., home, office, or customer site. In particular, virtual offices handle peripherals.

3.1 Cardinality-Based Feature Modeling (CBFM)

CBFM [6] extends FODA's feature diagrams with five main aspects: (1) cardinality which denotes the range of the feature clones that can be included at certain points, (2) feature groups which enable organizing features and defining how many group members can be selected at certain points (and, thus, they correspond to variation points), (3) attribute types indicating that an attribute value can be specified during configuration, (4) feature model references which enable splitting large feature models into smaller modules that are reference points to other features, and (5) OCL constraints enabling the description of additional constraints, such as "excludes" and "requires" dependencies.

In order to support addition rules, we used a blank feature with three dots as its name. This feature, which indicates openness, can be associated to feature groups that represent open variation points.

Figure 1 is a part of the VOF domain specification in CBFM. In this domain, two important concepts, which refer to separate feature diagrams that describe them (as depicted by the black triangles), are peripherals and the feasible operations on them (a). A peripheral is a device attached to a host computer and is dependent on the host. Each peripheral, which is specified as a feature group in (c), has its physical location, and may be characterized by its model and manufacturer, all of which are of type String. Possible variants of peripherals are fax machines, printers, scanners, and others. A fax machine, for example, is characterized by a phone number, possible paper sizes, and possible produced image qualities. Possible operations on peripherals, specified in (b), are document printing, conversion, scanning, and so on. The constraint specified in (d) implies that any VOF product that has a fax machine needs to be able to send and/or receive faxes.

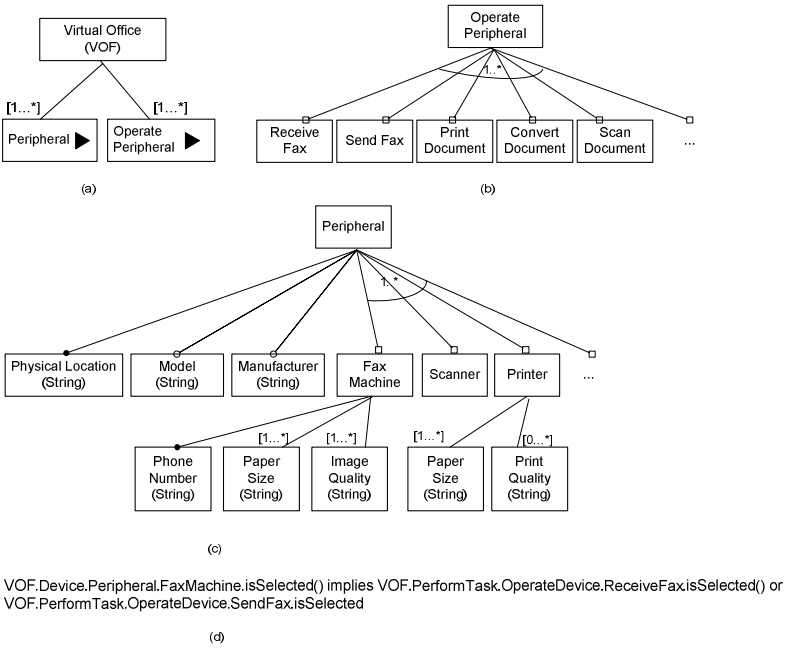


Fig. 1. A partial CBFM model of the VOF domain, concentrating on the peripheral capabilities

3.2 Application-Based Domain Modeling (ADOM)

ADOM defines a profile for specifying commonality and variability of core assets [22], which includes «multiplicity», «variation point», «variant», «requires», and «excludes» stereotypes. The *«multiplicity»* stereotype is used for specifying the range of product elements that can be classified as the same core element. Two tagged values, min and max, are used for defining the lowest and upper-most boundaries of that range. For clarity purposes, four commonly used multiplicity groups are defined on top of this stereotype: «optional many», where min=0 and max= ∞, «optional single», where min=0 and max=1, «mandatory many», where min=1 and max= ∞, and «mandatory single», where min=max=1. Nevertheless, any multiplicity interval constraint can be specified using the general stereotype «multiplicity min=m_1 max=m_2».

Each core asset element may define a variation point. This is done using the stereotype *«variation point»*, in addition to the «multiplicity» stereotype. A «variation point» stereotype has the following tagged values: (1) *open*, specifying whether the variation point is open or closed, i.e., whether product-specific variants that are not specified in the core asset can be added at this point or not, and (2) *card(inality)*, indicating the number of variant types need to be chosen for this variation point; common cardinalities are '1..1', '1..*', '0..1', and '0.*'. Note that there are differences between the «multiplicity» stereotype and the cardinality tagged values. A variation point, for example, can be optional (e.g., «optional many») while its cardinality specification is mandatory (e.g., '1..*'), indicating that this variation point may not be

included in a particular product, but if it is, then at least one of its variants (as specified in the core asset) have to be selected. Similarly, an open variation point can be mandatory (e.g., «mandatory many») while its cardinality specification is optional (e.g., '0..*'), indicating that this variation point has to be included in a particular product, but possibly use particular, product-specific variants (not specified in the core assets).

Each variant is specified using the *«variant»* stereotype, in addition to the «multiplicity» stereotype. A variation point and its variants should be of the same type (e.g., classes, attributes, associations, etc.). A variant is associated to the relevant variation point via inheritance relationships. When not applicable, i.e., for variation points and variants that are not classifiers, such as attributes and operations, the relationships between variants and variation points are specified using a tagged value, *vp*, associated to the «variant» stereotype; *vp* specifies the name of the corresponding variation point. Note that the same core element can be stereotyped by both «variation point» and «variant», enabling specification of hierarchies of variants.

Finally, two stereotypes are defined for determining dependencies between elements (and possibly between variation points and variants): «requires» and «excludes». A «requires» B implies that if A appears in a particular product artifact, then B should appear too. Similarly, A «excludes» B implies that if A is included in a particular product artifact, then B should not.

To exemplify the ADOM method, consider the partial VOF model depicted in Figure 3. The functional aspects of the peripheral operations are described in (a) via a use case diagram that describe the different variants of "Operate Peripheral", including "Send Fax", "Receive Fax", etc. The structural aspects of the peripherals are described in (b) via a class diagram that includes the different classes, their attributes, and relationships.

3.3 CBFM vs. ADOM

Although several works have been suggested for comparing feature-oriented and UML-based methods, e.g., [9], [19], and [27], none of them refer to comprehensibility of the resultant models to the stakeholders. Since both CBFM and ADOM can be used in early stages of core assets development, for specifying functional and structural aspects of product lines, we compare in this work their comprehensibility. However, to justify that these methods are comparable in terms of expressiveness,
Table 4 represents a mapping between CBFM and ADOM notations. Contrarily to ADOM, which explicitly enables specification of variability via variation points and variants, CBFM supports variability via feature groups and group cardinalities. Furthermore, in feature-orientation in general and in CBFM in particular, all features are similarly specified. This includes structural vs. functional features, classes vs. attributes, specializations vs. (physical) parts, and so on. The richness of UML in general and ADOM in particular is greater than that of feature-orientation, as the notation provides different symbols to the above aspects and enables specifying additional aspects, such as the involvement of different stakeholders in the system's functionality (in the form of actors and their participation relationships to different use cases). Despite these differences, the kernels of these two methods for specifying core assets are comparable and, thus, we empirically evaluate comprehension of their resultant models in order to better understand their benefits and limitations.

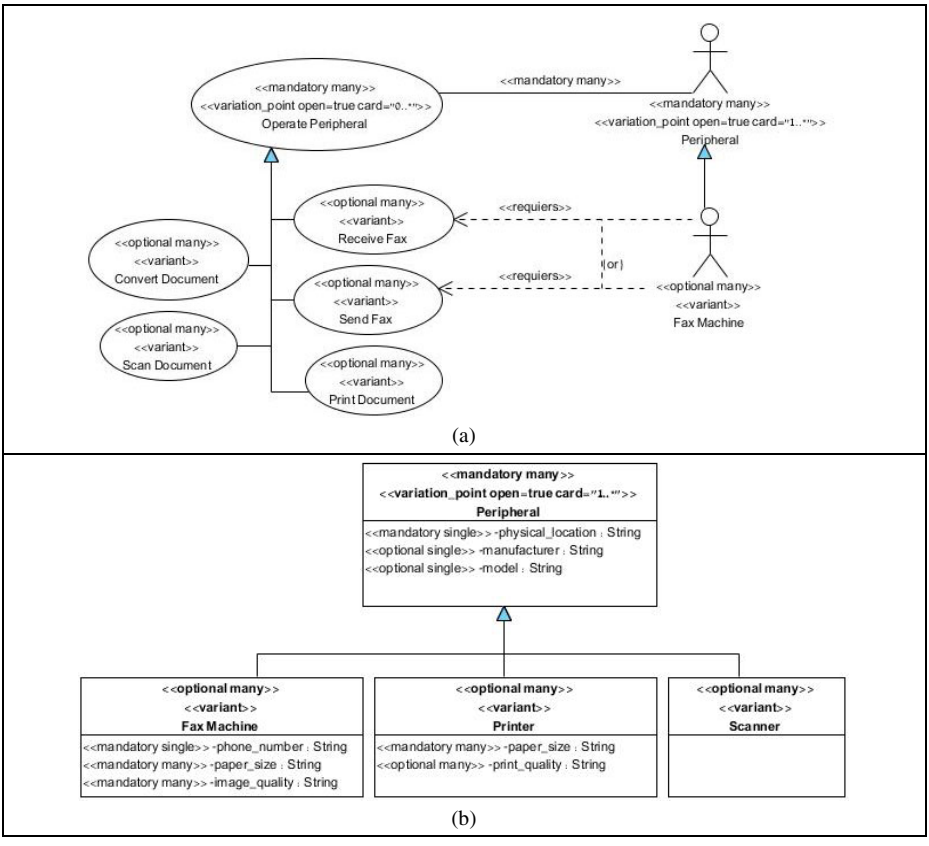


Fig. 2. A partial VOF model in ADOM: (a) A use case diagram describing "Operate Peripheral" and (b) A class diagram describing "Peripheral"

Table 4. CBFM vs. ADOM notations

CBFM	ADOM	CBFM	ADOM	CBFM	ADOM
F	«mandatory single»	Variant	«variant» «optional single»	m..n	«variation_point card='m..n'»
[1..m] F	«mandatory many»	Variant	«variant» «mandatory single»	requires	«requires»
F	«optional single»		«variation_point card='1..1'»	excludes	«excludes»
[0..m] F	«optional many»		«variation_point card='1..*'»	...	«variation_point open=true»

4 The Empirical Study for Comparing CBFM and ADOM

The conducted empirical study aimed at analyzing the comprehensibility of models specified in CBFM and ADOM to two main stakeholders in software product line

engineering: developers, who develop and maintain core assets or use them in order to create valid product artifacts, and product customers, who are involved in creating specific product artifacts that best fit their requirements and expectations. As core assets are an important means for communication between these stakeholders, the comprehensibility of their specification is important to be analyzed. Thus, we phrased the following null hypotheses that relate to the three research questions presented in Section 1: ($H_0$1) There is no difference in developer's comprehension of commonality aspects in CBFM and ADOM models (refers to RQ1); ($H_0$2) There is no difference in developer's comprehension of variability aspects in CBFM and ADOM models (refers to RQ1); ($H_0$3) There is no difference in product customer's comprehension of commonality aspects in CBFM and ADOM models (refers to RQ2); ($H_0$4) There is no difference in product customer's comprehension of variability aspects in CBFM and ADOM models (refers to RQ2); ($H_0$5) There is no difference in product customer's comprehension of commonality aspects in text specifications and model-based specifications in ADOM and CBFM (refers to RQ3); and ($H_0$6) There is no difference in product customer's comprehension of variability aspects in text specifications and model-based specifications in ADOM and CBFM (refers to RQ3).

Due to the difficulties to carry out such experiments on real customers and developers in industrial settings, we decided to experiment with the aforementioned methods with information systems students at the University of Haifa, Israel. Furthermore, Kitchenham et al. [17] argue that using students as subjects instead of software engineers is not a major issue, as long as the research questions are not specifically focused on experts. Since this is the case in our study, we turned to two different populations of students that we believe reflect the expected knowledge and skills from the potential stakeholders: first semester students in an IT introductory course played the role of product customers, while advance undergraduate and graduate information systems students in a domain engineering seminar played the role of developers. A pre-questionnaire, in which the subjects filled information regarding their modeling and analysis skills, their industrial experience, and their knowledge in different software development stages, in the compared methods, and in the specific experimented domain, confirmed our belief about the populations. The settings and the results of each experiment, as well as the threats to validity, are reported below.

4.1 Study Settings

The first experiment on developers (which will be called Exp1 from now on) took place in a seminar course named "Advanced Topics in Software Engineering", during the winter semester of the academic year 2010-2011. 18 subjects participated in this experiment and during the course they studied various domain engineering techniques, focusing on CBFM and ADOM and their ability to specify core assets. The study took place towards the end of the course as a class assignment, which worth up to 10 points of the students' final grades in the course. The students were equally divided into two groups of 9 students (each), according to the information we collected from the pre-questionnaires. Performing t-test on the average grades of the students in relevant courses, we found no significant difference between the groups ($t=1.29$, p-value=0.227). The students in the first group got a CBFM model of the VOF domain, part of which is depicted in Figure 1, and a dictionary of terms in the

domain, while the students in the second group got an ADOM model of the VOF domain, part of which is depicted in Figure 2, and the same dictionary of domain terms. The students in the two groups were asked to answer 15 true/false comprehension questions about the models they got and to provide full explanations to their answers. The questions in the two groups were similar, they refer to both commonality and variability aspects, and three experts checked that these questions can be answered via the two models separately. Furthermore, a single question could refer to several aspects and in these cases the question was counted several times in different categories, such that each low level category referred to 3-4 questions. Examples of questions in this study are:

> *A VOF application that handles fax machines may store for each fax machine only its serial number, phone number, paper size, and image quality. (A question that refers to variability specification of variants)*
>
> *A VOF application may not receive or send faxes, neither print, convert, or scan documents. (A question that refers to selection and addition rules of variability)*

The second experiment on customers (which will be called Exp2 from now on) took place in an introductory course called "Introduction to Information Technologies", during the winter semester of the academic year 2010-2011. 29 first semester students participated in this experiment. They were arbitrarily divided into three groups. The first group got the CBFM model of the VOF domain, the domain dictionary, and a short description of CBFM notation. The second group got similar documents for the ADOM method. Finally, the third group, which will be called the TEXT group, got a textual description of the VOF domain without any model. Here again we found no significant difference between the groups according to ANOVA test. Since the students were in their first semester of studies, we performed this test on their acceptance grades to the university and the results were F=0.957, p-value=0.4.

All students were asked to answer 10 true/false comprehension questions about products that they can have in the domain and to provide full explanations to their answers. The models and questions were similar to those in Exp1 and refer to both commonality and variability aspects. However, the number of questions and their wording were modified in order to fit to the knowledge and skills of the subjects and potential product customers. Still, the number of questions in each low level category was 3-4 (classifying single questions in several relevant categories).

4.2 Study Results

The answers in both experiments were checked according to a pre-defined solution. For each question, both the final answer (true/false) and the explanation were evaluated and received 0 (incorrect), 1 (correct), or 0.5 (partial, for the explanations only) points. We grouped the various questions according to the criteria they aim to check and present the average success percentages in Table 5. In each row the highest score is bolded (separately for each experiment), while in Exp 2, the higher score between CBFM and ADOM is underlined. The cells that present statistically significant (or borderline significant) results are presented in grey.

Table 5. The results achieved in the two experiments

Criterion		Exp 1 (Developers)				Exp 2 (Customers)				
		CBFM	ADOM	p-value	t(16)	CBFM	ADOM	TEXT	p-value	F(2, 26)
Commonality	Multiplicity	**76**	71	0.693	0.402	46	53	**76**	0.004	6.789
	(Inter-) Dependencies	**76**	74	0.884	0.148	32	46	**60**	0.116	2.349
	Overall Commonality	**76**	71	0.692	0.404	49	53	**71**	0.053	3.303
Variability	Variability Specification	71	**87**	0.029	-2.392	46	60	**67**	0.146	2.073
	Selection rules	70	**75**	0.755	-0.318	50	49	**70**	0.096	2.567
	Addition rules	74	**87**	0.421	-0.825	**67**	52	54	0.360	1.062
	Overall Variability	72	**83**	0.299	-1.073	50	56	**60**	0.427	0.879
Overall		73	**78**	0.559	-0.597	54	58	**62**	0.529	0.653

As can be seen, in Exp1, CBFM outperformed ADOM in commonality-related questions, while ADOM outperformed CBFM in variability-related questions. This outcome is reasonable, as feature-orientation concentrates on a common core and its possible configurations, while ADOM treats software products and product lines as belonging to two different abstraction levels and allows more variability among products that belong to the same line (e.g., via specialization and extension).

Not surprisingly most of the questions in Exp2 were better answered when the subjects had only textual description and no model. However, between ADOM and CBFM, ADOM outperformed CBFM in all commonality-related aspects and in variability specification questions, while CBFM outperformed ADOM in questions regarding selection and addition rules. Yet, in overall variability specification ADOM outperformed CBFM. Note that in the addition rules, CBFM outperformed both ADOM and TEXT groups, probably due to the clear notation of the three dots. Nonetheless, this notation was introduced to CBFM only for the experiment in order that CBFM will be comparable to ADOM in this category.

For getting better insights, we further performed statistical analysis on the results following the principles presented in [29]. Since a pre-check showed that the populations in both experiments were distributed normally, we conducted t-test for Exp1 and ANOVA test for Exp2. The statistical analysis of the results in Exp1 (presented in Table 5) shows that there is significant difference in variability specification in favor of ADOM (87 vs. 71). Thus, the second null hypothesis, $H_0 2$, can be rejected. In all other categories no significance was perceived, and, thus the first null hypothesis, $H_0 1$, cannot be rejected. Still, according to the presented averages, the overall comprehensibility in ADOM is better than that in CBFM (78 vs. 73, respectively).

The statistical analysis of Exp2 results, also presented in Table 5, shows significant difference in the multiplicity category in favor of the TEXT group and consequently borderline significance in the overall commonality category. Therefore, the fifth null

hypothesis, $H_0$5, can be rejected, while the sixth null hypothesis, $H_0$6, cannot be rejected. We further preformed Tukey's post-hoc test in order to determine which groups differ from each other. In the multiplicity category, there are significant differences between the TEXT group and both the ADOM group (p-value=0.026) and the CBFM group (p-value=0.004). In both cases, the differences are in favor of the TEXT group. The difference analysis between the ADOM group and the CBFM group, however, results in no significance (p-value=0.732), although ADOM outperformed CBFM in this category (53 vs. 46). In the overall commonality category, no significance was perceived, although the difference between the TEXT group and the CBFM group in this category is borderline (p-value=0.058). Thus, the third and the forth null hypotheses, $H_0$3 and $H_0$4, cannot be rejected.

4.3 Threats to Validity

The main threats to validity are of course the small number of the subjects in each experiment (18 in Exp1 and 29 in Exp2), the relatively simple tasks and models, and the specific selected SPLE methods. To overcome these limitations, the following actions were taken. The students in Exp 1 were pre-assigned to groups in order to perform similarly capable groups. For increasing their motivation, they got up to 10 points for their performance in the assignment and thus tried to do their best. In Exp2, the division into groups was arbitrarily, but no statistical difference has been found afterwards. We carefully chose the domain and specified the corresponding models so that they will refer to different domain engineering-related challenges. Three experts checked the models and their equivalent expressiveness before the experiment. However, we aim at keeping the models simple, yet realistic, so that the subjects will be able to respond in reasonable time. Furthermore, we used a relatively small number of questions in order to avoid subjects' exhaustion. Yet, we chose questions that could be classified to different investigated categories, but this may increase the complexity of the individual questions. Finally, we conducted a comparative analysis between existing feature-oriented and UML-based methods. The methods used in the experiment were selected based on this analysis, which was presented in Section 2.

5 Summary and Future Work

Comprehension of core assets is important when dealing with software product line engineering (SPLE). In this paper, we checked comprehension of both commonality and variability issues in core asset specifications. In particular, we compared two methods from different leading modeling paradigms: CBFM, which is a feature-oriented method, and ADOM, which is based on a UML profile. Both methods enable specifying functional and structural requirements in the form of feature diagrams in CBFM and UML use case and class diagrams in ADOM. We found that despite of the common assumption that feature-orientation is simpler and thus more comprehensible [14], [24], ADOM, which actually proposes an integrated profile to software product line engineering, is not less comprehensible than feature-oriented models. Furthermore, in terms of specifying the allowed variability, ADOM may offer better aids to developers than feature-oriented methods in general and CBFM in particular.

Additionally, as opposed to feature-orientation that focuses on early development stages of requirements engineering, UML-based methods encompass the entire development lifecycle and software engineers and developers are more familiar with their notation. An evidence of this claim can be found in works that extend feature-orientation to the design and implementation phases introducing UML concepts, e.g., [4] and [24]. Finally, and with respect to text specifications, significance has been found only in commonality related issues.

Only further studies may confirm or disconfirm whether our results can be generalized to more experienced subjects, more complicated models and tasks, and other modeling methods. In particular, we intend to check how the comprehensibility of models in these methods influences (if at all) the creation of product artifacts and modification of core assets. We also plan to extend the checked expressiveness criteria, e.g., to binding time and reuse mechanisms.

References

1. Alves de Oliveira, E., Gimenes, I.: A Variability Management Process for Software Products Lines. In: The 2005 conference of the Centre for Advanced Studies on Collaborative Research, pp. 225–241 (2005)
2. Bachmann, F., Clements, P.C.: Variability in Software Product Lines. Technical Report CMU/SEI-2005-TR-012 05tr012 (2005), `http://www.sei.cmu.edu/library/abstracts/reports/05tr012.cfm`
3. Clauß, M.: Generic Modeling using UML extensions for variability. In: OOPSLA 2001 Workshop on Domain Specific Visual Languages (2001)
4. Clauß, M.: Modeling variability with UML. In: GCSE 2001 Young Researchers Workshop (2001)
5. Coriat, M., Jourdan, J., Fabien, B.: The SPLIT method: building product lines for software-intensive systems. In: The 1st Conference on Software Product Lines: Experience and Research Directions, pp. 147–166 (2000)
6. Czarnecki, K., Kim, C.H.P.: Cardinality-based Feature Modeling and Constraints: A Progress Report. In: OOPSLA Workshop on Software Factories (2005)
7. Czarnecki, K., Eisenecker, U.W.: Generative Programming: Methods, Tools, and Applications. Addison-Wesley, Reading (2000)
8. Eriksson, M., Börstler, J., Borg, K.: The PLUSS approach - domain modeling with features, use cases and use case realizations. In: Obbink, H., Pohl, K. (eds.) SPLC 2005. LNCS, vol. 3714, pp. 33–44. Springer, Heidelberg (2005)
9. Frakes, W.B., Kyo, K.: Software Reuse Research: Status and Future. IEEE Transactions on Software Engineering 31(7), 529–536 (2005)
10. Gomaa, H.: Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. Addison-Wesley Professional, Reading (2004)
11. Griss, M., Favaro, J., d'Alessandro, M.: Integrating Feature Modeling with the RSEB. In: The 5th International Conference on Software Reuse (ICSR 1998), pp. 76–85 (1998)
12. Halmans, G., Pohl, K.: Communicating the Variability of a Software-Product Family to Customers. Software and Systems Modeling 2(1), 15–36 (2003)
13. Halmans, G., Pohl, K., Sikora, E.: Documenting application-specific adaptations in software product line engineering. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 109–123. Springer, Heidelberg (2008)

14. Jaejoon, L., Dirk, M.: Feature-Oriented Variability Management in Product Line Engineering. Communications of the ACM 49(12), 55–59 (2006)
15. Kang, K. Lee, J.: FOPLE – Feature-Oriented Product Line Software Engineering: Principles and Guidelines, Pohang University of Science and Technology (2002)
16. Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, A.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University (1990)
17. Kitchenham, B.A., Lawrence, S., Lesley, P., Pickard, M., Jones, P.W., Hoaglin, D.C., Emam, K.E.: Preliminary Guidelines for Empirical Research. IEEE Transactions on Software Engineering 28(8), 721–734 (2002)
18. Kyo, C.K., Sajoong, K., Jaejoon, L., Kijoo, K., Euiseob, S., Moonhang, H.: FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures. Annals of Software Engineering 5(1), 143–168 (1998)
19. Matinlassi, M.: Comparison of Software Product Line Architecture Design Methods: Comparison of Software Product Line Architecture Design Methods: COPA, FAST, FORM, KobrA and QADA. In: The 26th International Conference on Software Engineering (ICSE 2004), pp. 127–136 (2004)
20. Morisio, M., Travassos, G., Stark, M.: Extending UML to support Domain Analysis. In: The 15th IEEE International Conference on Automated Software Engineering, p. 321 (2000)
21. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer, Heidelberg (2005)
22. Reinhartz-Berger, I., Sturm, A.: Utilizing Domain Models for Application Design and Validation. Information and Software Technology 51(8), 1275–1289 (2009)
23. Riebisch, M., Böllert, K., Streitferdt, D., Philippow, I.: Extending Feature Diagrams with UML Multiplicities. In: The 6th Conference on Integrated Design and Process Technology, IDPT 2002 (2002)
24. Robak, S., Franczyk, B., Politowicz, K.: Extending the UML for modeling variability for system families. International Journal of Applied Mathematics and Computer Science 12(2), 285–298 (2002)
25. Sinnema, M., Deelstraa, S.: Classifying Variability Modeling Techniques. Information and Software Technology 49(7), 717–739 (2007)
26. Svahnberg, M., Van Gurp, J., Bosch, J.: A Taxonomy of Variability Realization Techniques. Software Practice & Experience 35(8), 705–754 (2005)
27. Trigaux, J.C., Heymans, P.: Modeling Variability Requirements in Software Product Lines: A Comparative Survey. Technical Report of PLENTY project, Institut d'Informatique FUNDP, Namur, Belgium (2003)
28. Webber, D., Gomaa, H.: Modeling Variability in Software Product Lines with Variation Point Model. Science of Computer Programming 53, 305–331 (2004)
29. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A.: Experimentation in Software Engineering – An Introduction. Kluwer Academic Publishers, Dordrecht (2000)
30. Ziadi, T., Hëlouët, L., Jézéquel, J.-M.: Towards a UML profile for software product lines. In: van der Linden, F.J. (ed.) PFE 2003. LNCS, vol. 3014, pp. 129–139. Springer, Heidelberg (2004)

Individual Differences and Conceptual Modeling Task Performance: Examining the Effects of Cognitive Style, Self-efficacy, and Application Domain Knowledge

Manpreet K. Dhillon and Subhasish Dasgupta

George Washington University
dhillonm@gwu.com, dasgupta@gwu.edu

Abstract. In information systems development, conceptual modeling, which includes both data modeling and process modeling, is the most effective technique for depicting and sharing an understanding of the functional capabilities and limitations of the product/ system/ service design. The quality of conceptual models depends on a number of factors. This research focused on attributes of the modeler and specifically examined how an individual's cognitive style, task self-efficacy, and knowledge of application domain impact the quality of two types of conceptual models: data models and process models. Results of the research revealed that an individual's cognitive style may relate to conceptual model quality. In addition, the research showed that self-efficacy may be a determinant of model quality. Application domain knowledge did not appear to play a role in quality of models produced by the participants in this study.

Keywords: individual differences, conceptual modeling, modeling performance.

1 Introduction

In just about all methodologies for systems development, assuring the quality of conceptual models is extremely important. Research has shown that half of the errors identified in the development stage were attributed to inaccurate or incomplete requirements specification [21]. According to a study conducted by Standish Group in 2002, in which over 13,000 information technology (IT) projects were analyzed, close to half of the projects do not document required features or functions on product release. Design errors can be very expensive, and early error detection can save enormous expense [11]. It is estimated that 55 to 85 percent of all software errors are attributable to undetected conceptual model design errors [11]. As a result, interest is growing in conceptual modeling research. Scholars cite several reasons to focus attention on this area of information systems development process including the emergence of object-oriented design approach, the challenges faced in eliciting requirements, and the desire to reuse software components [34, 37].

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 483–496, 2011.

2 The Research Problem

This study aimed to address this research gap by answering the following question: *How do individual differences relate to conceptual modeling task performance?* More specifically, this research focuses on how individual differences related to application domain knowledge, self-efficacy beliefs, and cognitive style impact conceptual modeling task performance. The approach adopted here applies three constructs; self efficacy (derived from Social Cognitive Theory (SCT)), cognitive style, (derived from Psychological Type Theory), and application domain knowledge (based on recent empirical literature) to investigate the influence each construct has on the performance of two types conceptual modeling tasks.

3 Theoretical Framework

A conceptual research model linking the various individual differences variables of interest and task performance is presented next. The model aims at explaining differences in performance between individuals, and posits that individual differences in application domain knowledge, cognitive style, and self-efficacy relate to conceptual modeling task performance. As a result, these differences are expected to exhibit variations in levels of task performance.

The research model is comprised of three factors, which theory and previous empirical research suggest may influence individual performance on information systems tasks and specifically on data modeling and process modeling tasks. These factors, represented as independent variables in the model, are application domain experience, cognitive style, and self-efficacy. The model's dependent variable is conceptual modeling task performance. The model aims to explain differences in individual task performance based on differences in levels of the application domain knowledge, different cognitive styles, and levels of task self-efficacy. The next section provides a brief summary of each of the constructs in the research model, and a set of corresponding hypotheses to be tested is presented.

3.1 Application Domain Knowledge/Experience – Performance Relationship

Application Domain knowledge is a powerful determinant of cognitive performance [12]. Research indicates that individuals with greater domain experience/familiarity have an advantage in problem solving beyond that conferred by well-developed knowledge structures [35]. One of the primary goals of requirements gathering during information systems analysis is the conveying and understanding the application domain [10]. Conceptual models are used to describe some aspect of the domain for the purposes of understanding and communication. Gemino and Wand [10] cite 4 roles that conceptual models play in developing domain understanding: (1) aiding a person's own understanding of a domain, (2) communicating domain details between stakeholders, (3) communicating domain details to systems designers, and (4) documenting the domain for future reference. Findings from several studies support the relationship between application domain knowledge and task performance [17, 31, 12, 33]. The results of these studies provide support for the view that individual

differences in domain knowledge may account for variations in performance of tasks related to that domain. Based on the literature reviewed, it is reasonable to expect that application domain knowledge/experience may improve conceptual modeling task performance. Conceptual modeling requires an understanding or knowledge of the domain in order to document key concepts and relationships. Modelers who have more experience with the domain are likely to perform better on conceptual modeling tasks than modelers who have less or no experience/knowledge of the domain. Specifically, one might expect that an individual will perform better in a conceptual modeling task in a familiar domain than an unfamiliar domain. This leads to the following hypothesis:

> *H1: Conceptual modeling task performance will be higher for familiar domains than for unfamiliar domains.*

3.2 The Cognitive Style – Performance Relationship

Typically, a person's mode of information acquisition combined with his/her information processing mode constitutes their cognitive style [25]. Thus, four distinct cognitive styles result from these two dimensions: (1) sensory-thinker (ST); (2) intuitive-thinker (NT); (3) sensory-feeler (SF); and (4) intuitive-feeler (NF). There are certain characteristics which relate to each of the 4 styles. ST types are typically concerned with technical details, engage in logical orderly processing of data and information, look for facts oriented with logic, are orderly and precise, and generally tend to display low tolerance for ambiguity. NT types tend to synthesize and interpret, look for ideas rather than facts associated with logic, emphasize understanding, are generally objective, impersonal, and idealistic, and tackle ill-defined and abstract situations. SF types are more subjective in their decision making, engage in more open communication, and are interested in facts associated with people rather than logic. Finally, NF types tend to be more creative, insightful, futuristic, and interested in ideas oriented with people rather than logic.

Data modeling and process modeling creation tasks are well-structured tasks that require attention to detail, logic, categorization of concepts, and organization. Based on the previous discussion of strengths and weaknesses, cognitive characteristics, and sources of work stress associated with the different psychological types, it is reasonable to expect that individual performance on data modeling and process modeling tasks may vary according to one's cognitive style. This leads to the second hypotheses.

> *H2: Conceptual modeling task performance will vary by cognitive style. Specifically, STs will perform higher than the other three cognitive style groups.*

3.3 The Self-efficacy – Performance Relationship

Researchers have noted that raising self-efficacy in information systems students may be an effective way to maximize performance outcomes and prepare them for on-the-job success [30, 16]. Despite the fact that it is a widely accepted and empirically validated theory of individual behavior, and believed to have strong influence on performance outcomes, Rozell and Gardner [29] note that Social Cognitive Theory

"has been virtually ignored in information systems research" [4]. According to social cognitive theory, students' self-efficacy beliefs – their judgments of confidence to perform academic tasks or succeed in academic activities – predict their subsequent capability to accomplish those tasks or succeed in those activities [28]. Self-efficacy is most relevant to complex tasks and conceptual modeling is often perceived as complex task by novice modelers. Because of the broad applicability of Social Cognitive Theory on performance outcomes, self-efficacy is hypothesized to have a positive relationship to conceptual modeling task performance. In other words, it is hypothesized that individuals who have a higher degree of task self-efficacy will perform better than individuals with a lower level of task self-efficacy.

H3: Self-Efficacy is positively related to conceptual modeling task performance.

4 Research Methodology

This study employed survey methodology in conjunction with performance on data modeling and process modeling quizzes. This study required participants to complete survey instruments in order to measure the independent variables. Dependent variables were assessed according to performance on a data modeling and process modeling tasks.

Variable Operationalization

Three independent variables are used in the research design: (1) Application domain experience, (2) cognitive style, and (3) task self-efficacy. Application domain familiarity was controlled for by selecting one familiar domain and one unfamiliar domain. Based on informal feedback from students enrolled in the same course in a previous semester, two application domains were selected for this study: (1) Social Networking (Facebook) domain and (2) a Medical Billing domain. It was expected that participants would be highly familiar with the Facebook domain and less familiar with the medical billing domain. A further check on application domain familiarity was made by asking participants to rate their level of familiarity with certain terms and processes related to both domains. Data for the two other independent variables, self-efficacy and cognitive style, were captured through the use of survey instruments, which are described in more detail later. The dependent variable, 'conceptual modeling task performance', is operationalized as model quality. Model quality measures were adopted from previous research. Model quality has been identified as an important topic in current conceptual modeling research. Although there are a number of frameworks proposed in the literature for evaluating the quality of conceptual models, the framework proposed by Lindland et al. [27] was be used. This framework is based on semiotic theory, and consists of three sub-categories of model quality: syntactic, semantic, and pragmatic. This framework has been used in recent studies [21, 26], and can be applied to different types of modeling artifacts, which was important in this study since both data modeling and process modeling tasks were examined. Overall performance was based on quality ratings in three subcategories: syntactic quality, semantic quality, and pragmatic quality.

4.1 Self-efficacy Measure

Self-Efficacy is a task specific construct and theory requires that its measurement is also confined to the particular task being investigated [2]. Bandura recommended that self-efficacy measures include both self-efficacy magnitude, and strength. Lee and Bobko [20] analyzed common ways in which self-efficacy is operationalized. They concluded that a composite measure of self-efficacy magnitude and strength showed the highest convergent and divergent validity. Self-efficacy magnitude is determined by asking participants to indicate whether or not they can execute the task (yes or no). Self-Efficacy strength is determined by asking the respondents to what degree they are confident they can execute each task.

4.2 Application Domain Knowledge/Experience Measure

Application domain knowledge was operationalized at two levels: familiar and unfamiliar. Based on informal focus group information received from a group of students enrolled in the same course in a previous semester, the researcher pre-selected two domains to utilize in this study. One domain represented a business area which study participants would be expected to be very familiar with and the second domain represented a business are which the participants are likely to have little familiarity/experience with. The familiar domain chosen for this study is the popular social networking site, Facebook. The unfamiliar domain is Medical Billing. Participants completed the application domain questionnaire prior to completing the conceptual modeling tasks.

4.3 Cognitive Style

More than five decades ago, Katherine Briggs and Isabel Myers began work on an instrument to operationalize Jung's (1921/1971] theory of psychological types, the Myers-Briggs Type Indicator (MBTI). A number of instruments have been used to measure cognitive style. Examples include the Kirton Adaptor Innovators (KAI) instrument [18], and the Cognitive Style Index (CSI) [1]. The most common measure of cognitive style, however, is the Myers-Briggs Type Indicator [23, 38]. The MBTI is extremely popular in industry, and it is administered to more than three million people a year (more than 40 percent of these users work in major corporations) [8]. Researchers have argued that studies based on such widely utilized instruments have more relevance to organizations than studies based on more seldom-applied measures [9]. There are more than 4,000 published sources regarding the MBTI's validity and reliability. Additionally, even though other cognitive style instruments are shorter (such as CSI and KAI), they are highly correlated with the MBTI.

In order to measure cognitive style, the MBTI: Form M was used. Although only two of the four dimensions of the MBTI are used as cognitive style measures (Sensing-Intuition and Thinking-Feeling), respondents were asked to complete the entire MBTI-Form M.

4.4 Description of Tasks and Task Performance Measurement

A total of four modeling tasks were used in this research: two process model creation tasks (one in the familiar domain and one in the unfamiliar domain) and two data model creation tasks (on in the familiar domain and one in the unfamiliar domain). The tasks were developed with the help of an informal focus group of undergraduate students who had previously taken the information systems course and were familiar with both types of tasks. The dependent variable, task performance was operationalized as model quality.

5 Research Protocol

This section addresses the procedures for data collection. It discusses the protocol governing subject participation and the incentives used to promote voluntary participation. Finally, the safeguards that have been implemented to protect subject privacy and to secure data are discussed briefly.

5.1 Study Procedures

Prior to data collection, students enrolled in an undergraduate introductory management information systems class received an informational letter of consent, stating that the general purpose of the study is to "understand how individual differences relate conceptual modeling task performance". They were assured that the subject material of the research is consistent with course objectives, and while completion of the research related survey instruments is voluntary, it is anticipated that there would knowledge benefits from participating in the study. In order to encourage participation, students who completed all survey instruments received 5 extra credit points towards a database quiz not associated with the research. The database quiz was administered after the all study materials were collected. The process modeling and data modeling tasks were scheduled quizzes in this course for which students received grades which counted toward their final course grade. Therefore, an added incentive for completing the tasks was not needed. Students, who choose not to participate in the survey, and did not complete the survey instruments, had the option of completing an alternative assignment related to conceptual modeling in order to earn the extra credit points. A total of 132 students completed all study requirements and were included in the final data analysis.

6 Data Analysis

A combination of regression, analysis of variance, and both paired and independent samples t-tests were used analyze the data in this study. Variables included both categorical and interval data. In this section we provide analysis and results for each hypothesis. The results for the two conceptual model types (data and process) will be discussed separately. First, the analysis and findings for the data model tasks is presented.

6.1 Data Model Results

Results of the hypotheses testing for the data model tasks are presented below.

HYPOTHESIS 1: Task performance in the familiar domain will be higher than task performance in the unfamiliar domain

In order to test this hypothesis, a paired samples t-test was conducted. Results are presented below.

Table 1. A paired samples t-test - Descriptive

Descriptives			
	Mean	**N**	**Standard Deviation**
Familiar Domain Performance	16.75	132	2.569
Unfamiliar Domain Performance	16.51	132	3.109

Table 2. A paired samples t-test - Paired Samples Correlations

Paired Samples Correlations		
	Correlation	**Sig.**
Familiar Domain Performance & Unfamiliar Domain Performance	.738	.000

Data Model performance scores in the familiar domain (Facebook) and unfamiliar domain (Patient Billing) are highly correlated. This suggests that if an individual obtains a high score in one domain, they are likely to obtain a high score in the other domain.

Table 3. A paired samples t-test result

Paired Samples Test		
	T-Value	**Sig. (2-tailed)**
Familiar Domain Performance & Unfamiliar Domain Performance	1.264	.208

Based on the results obtained from the paired-samples t-test presented above, we fail to reject the null and therefore cannot conclude that there is a statistically significant difference between task performance in the familiar (Facebook) domain and task performance in the unfamiliar (Medical Billing) domain.

HYPOTHESIS 2: Task Performance will vary by cognitive style. Specifically, STs are expected to perform higher than the other three groups.

A One-way analysis of variance (ANOVA) was performed to test this hypothesis. The ANOVA test results are presented in the table below.

Table 4. A One-way analysis of variance (ANOVA)

ANOVA						
		Sum of Squares	**df**	**Mean Square**	**F**	**Sig.**
Familiar Domain Performance	Between Groups	47.348	3	15.783	2.472	.065**
	Within Groups	817.275	128	6.385		
	Total	864.623	131			
Unfamiliar Domain Performance	Between Groups	41.744	3	13.915	1.454	.230
	Within Groups	1224.795	128	9.569		
	Total	1266.539	131			

** p-value < .10.

Based on the ANOVA table, there is a statistically significant difference in data model performance in the familiar domain. The results obtained from the Scheffe post-hoc analysis indicate which cognitive style pair mean scores are significantly different.

Table 5. The results obtained from the Scheffe post-hoc analysis

Scheffe Results			
	(j) Sensory-Feeler	**Intuitive-Thinker**	**Intuitive-Feeler**
(i) Sensory-Thinker	-1.834*	-1.229	-1.154
Sensory-Feeler		.605	.680
Intuitive-Thinker			.075

* p-value < .05.

There is a statistically significant difference in data model performance in the familiar domain between Sensory-Thinkers and Sensory-Feelers. Specifically, Sensory-Feelers on average score close to 2 points higher (out of a total of 21 points) than Sensory Thinkers. Although it was not part of the hypotheses testing, an additional set of tests were conducted to compare task performance between each of the MBTI dichotomous pairs. The interest in running these tests was to observe if there were differences in performance between (1) Extraverts and Introverts, (2) Sensing and Intuition, (3) Thinking and Feeling, and (4) Judging and Perceiving.

Four separate independent samples t-tests were performed in order to obtain this information. A summary of the results is provided in the table below.

Table 6. A summary of the t-tests results

		Extravert vs. Introvert	Sensing vs. Intuition	Thinking vs. Feeling	Judging vs. Perceiving
Familiar Domain Performance	t-value	-.827	-1.520	-.1.64	1.143
	(sig.)	(.140)	(.131)	(.103)	(.255)
Unfamiliar Domain Performance	t-value	1.02	-1.89	-.814	.111
	(sig.)	(.310)	(**.060**)**	(.417)	(.912)

* p-value < .05 level ** p-value < .10 level.

The only significant difference between the pairs was found between Sensing and Intuition in the unfamiliar domain. The results suggest that on average, Intuitive individuals scored 1.05 points higher than Sensing individuals on the data model task in the unfamiliar domain.

HYPOTHESIS 3: Self-efficacy will be positively related to task performance.

Regression analysis was conducted to test this hypothesis.

Table 7. Regression analysis

Regression Results					
Dependent Variable	**R-squared**	**F-value**	**Independent Variable**	**Coefficient**	**T-value**
Familiar Domain Score	.193	31.065	Data Model Self Efficacy	.439	5.574*
Unfamiliar Domain Score	.131	19.613	Data Model Self Efficacy	.362	4.429*

* p-value < .01 level.

Based on the results of the regression analysis, there is a statistically significant relationship between data model self efficacy and data model task performance in both the familiar and unfamiliar domains. Therefore we reject the null and can confidently conclude that within the sample population of this study, higher perceptions of task self-efficacy are directly related to higher performance in conceptual data model creation tasks.

6.2 Process Model Results

HYPOTHESIS 1: Task performance in the familiar domain will be higher than task performance in the unfamiliar domain

Results of the paired-samples t-test are provided in the tables below.

Table 8. Paired Samples Correlations

Paired Samples Correlations		
	Correlation	**Sig.**
Familiar Domain Performance & Unfamiliar Domain Performance	.537	.000

Process Model performance scores in the familiar domain (Facebook) and unfamiliar domain (Medical Billing) are highly correlated. This suggests that if an individual obtains a high score in one domain, they are likely to obtain a high score in the other domain.

Table 9. Paired Samples Test

Paired Samples Test		
	T-Value	**Sig. (2-tailed)**
Familiar Domain & Unfamiliar Domain Perform	-1.235	.219

Based on the results obtained from the paired-samples t-test presented above, we fail to reject the null and therefore cannot conclude that there is a statistically significant difference between process model performance scores in the familiar (Facebook) domain and performance scores in the unfamiliar (Medical Billing) domain.

HYPOTHESIS 2: Task Performance Scores will vary by cognitive style. Specifically, STs are likely to perform higher than the other three groups.

The ANOVA results are presented below.

Table 10. ANOVA results

ANOVA						
		Sum of Squares	df	Mean Square	F	Sig.
Familiar Domain Performance	Between Groups	10.946	3	3.649	1.764	.157
	Within Groups	264.821	128	2.069		
	Total	275.767	131			
Unfamiliar Domain Performance	Between Groups	19.735	3	6.578	2.047	.111
	Within Groups	411.286	128	3.213		
	Total	431.021	131			

Based on the ANOVA table, we do not find a statistically significant difference in process model performance (in either domain) based on the four cognitive style groups. Therefore we fail to reject the null and cannot conclude that there is a statistically significant difference in process model task performance based on cognitive style.

The additional set of tests conducted to compare task performance between each of the four MBTI dichotomous pairs is presented below.

Table 11. Task performance between each of the four MBTI dichotomous pairs

		Extravert vs. Introvert	Sensing vs. Intuition	Thinking vs. Feeling	Judging vs. Perceiving
Familiar Domain Perf	t-value (sig.)	-.443 (.659)	.153 (.879)	-.950 (.344)	-1.997 (.048)
Unfamiliar Domain Perf	t-value (sig.)	1.71 (.090)	-2.00 (**.047)***	-.309 (.758)	-2.04 (**.044)***

* p-value < .05 ** p-value < .10.

A significant difference was found between Sensing and Intuition in the unfamiliar domain. On average, Intuitive individuals scored approximately 1 point higher than Sensing individuals on the process model task in the unfamiliar domain.

A significant difference was also found between Judging and Perceiving in the unfamiliar domain. On average, Perceiving individuals scored approximately 1 point higher than Judging individuals on the process model task in the unfamiliar domain.

HYPOTHESIS 3: Self-efficacy will be positively related to task performance.

Results of the regression analysis are presented below.

Table 12. Results of the regression analysis

REGRESSION					
Dependent Variable	**R-squared**	**F-value**	**Independent Variable**	**Coefficient**	**T-value**
Familiar Domain Performance	.118	17.440	Process Model Self Efficacy	-.344	-4.176 *
Unfamiliar Domain Performance	.110	16.053	Process Model Self Efficacy	-.332	-4.007 *

* p-value < .05.

Based on the results of the regression analysis, there is a statistically significant relationship between process model self efficacy and process model task performance in both domains. Specifically, there is a negative relationship between process model

self-efficacy and process model task performance. This suggests that the more confident an individual is in his/her process modeling skills, the lower his/her performance score will be.

7 Discussion and Implications

Although subjects in this research study were clearly more familiar with the Facebook domain than the Medical Billing domain, the quality of the models produced based on the Facebook domain did not differ significantly from the quality of the models produced in the less familiar Medical Billing domain. Similar results were found in Khatri et al's [17] study which examined the role of application domain knowledge and performance in conceptual model schema understanding tasks. The authors suggest that because the tasks involved extracting knowledge that was represented directly in the conceptual schema, the added knowledge of the application domain was not necessary for successful task performance. This explanation may hold true in this research as well. In the current study, the task required individuals to develop a model (data model or process model) based on business requirements provided to them. It was not necessary for individuals to seek further application domain related information in order to complete the modeling tasks. It is quite possible that the effect of application domain knowledge would have been significant if part of the task requirements included gathering and summarizing the business requirements to be modeled for each application domain.

A second objective of this study was to investigate whether individual differences in cognitive style can be linked to conceptual modeling task performance variations. However, it should be noted that the number of SFs were significantly lower than the other groups: specifically there were 13 SFs compared to more than 30 in each of the other three cognitive style types. It was anticipated that Sensory-Thinkers (STs) would outperform Sensory-Feelers (SFs). However, the exact opposite occurred in this study, and although there was only a slight difference in performance levels, STs actually performed lower than the three other cognitive style groups. Overall, the results suggest that cognitive style is not a predictor of conceptual modeling task performance for novice modelers.

The findings related to data model task performance support Ryan's et al [30] findings, and provide further evidence to support Bandura's [2]claim, based on Social Cognitive Theory, that self-efficacy beliefs predict academic outcomes. The negative relationship found between self-efficacy and the process modeling task requires further investigation, as it appears that this is one of very few studies in the literature that have examined this relationship.

A similar significant positive relationship between task self-efficacy and process modeling performance was anticipated. Therefore, the negative relationship between process model self-efficacy and task performance was somewhat unexpected. A review of literature did not reveal any previous studies which examined this relationship. Similar studies should be conducted to validate these findings, as the negative correlation may have been due to other factors specific to this study. For example, in the current study it is possible that individuals over-estimated their confidence in the process modeling. During and after lecture sessions on both data and process

modeling, some students expressed that they thought process modeling was considerably easier than data modeling. However, upon completion of the research tasks, some individuals commented that they felt that the process modeling tasks were more difficult than the data modeling tasks.

In summary, we believe that this research has made a significant contribution to research, and we encourage additional research in the area.

References

1. Allison, C.W., Hayes, J.: The Cognitive Style Index: A Measure of Intuition-Analysis for Organizational Research. Journal of Management Studies 33(1), 119–135 (1996)
2. Bandura, A.: Social foundations of thought and action: a social cognitive theory. Prentice-Hall, Englewood Cliffs (1986)
3. Bolloju, N., Leung, F.S.K.: Assisting Novice Analysts in Developing Quality Conceptual Models. Communications of the ACM 49(7), 108–112 (2006)
4. Compeau, D.R., Higgins, C.A.: A social cognitive theory perspective on individual reactions to computing technology. In: Proceedings of the Twelfth International Conference on Information Systems, New York (1991)
5. Compeau, D.R., Higgins, C.A.: Application of social cognitive theory to training for computer skills. Information Systems Research 6, 118–143 (1995a)
6. Compeau, D.R., Higgins, C.A.: Computer self-efficacy: development of a measure and initial test. MIS Quarterly 19, 189–211 (1995b)
7. Franco, L., Meadows, M.: Exploring new directions for research in problem structurin methods: on the role of cognitive style. Journal of the Operational Research Society 58, 1621–1629 (2007)
8. Gardner, W.L., Martinko, M.J.: Using the Myers-Briggs Type Indicator to Study Managers: A Literature Review and Research Agenda. Journal of Management 22(1), 45–83 (1996)
9. Garfield, M.J., Taylor, N.J., et al.: Research Report: Modifying Paradigms - Individual Differences, Creativity, Techniques, and Exposure to Ideas in Group Idea Generation. Information Systems Research 12(3), 322–333 (2001)
10. Gemino, A., Wand, Y.: A Framework for Empirical Evaluation of Conceptual Modeling Techniques. Requirements Engineering 9, 248–260 (2004)
11. Gerard, G.J.: The REA Pattern, Knowledge Structures, and Conceptual Modeling Performance. Journal of Information Systems 19(2), 57–77 (2005)
12. Hambrick, D.Z., Engle, R.W.: Effects of Domain Knowledge, Working Memory Capacity, and Age on Cognitive Performance: An Investigation of the Knowledge-Is-Power Hypothesis. Cognitive Psychology 44, 339–387 (2002)
13. Isaksen, S.G., Lauer, K.J., et al.: An Examination of the Relationship Between Personality Type and Cognitive Style. Creativity Research Journal 15(4), 343–354 (2003)
14. Jahng, J., Jain, H., et al.: Personality Traits and effectiveness of presentation of product information in e-business systems. European Journal of Information Systems 11, 181–195 (2002)
15. Jones, W.P.: Computer Use and Cognitive Style. Journal of Research on Computing in Education 26(4), 514–523 (1994)
16. Jordan, D., Whalen, G., Bell, B., McKeown, K., Feiner, S.: An evaluation of automatically generated briefings of patient status. In: Medinfo. 2004 (2004)
17. Khatri, V., Vessey, I., et al.: Comprehension of conceptual schemas: Exploring the role of application and IS domain knowledge. Information Systems Research 17(1), 81–99 (2006)

18. Kirton, M.: Adaptors and innovators: A description and measure. Journal of Applied Psychology 61, 622–629 (1976)
19. Kovar, S.E., Ott, R.L., et al.: Personality preferences of accounting students: a longitudinal case study. Journal of Accounting Education 21, 75–94 (2003)
20. Lee, C., Bobko, P.: Self-efficacy; Moods-Psychology; Goal-Psychology; Psychology-Methodology. Journal of Applied Psychology 79, 364–369 (1994)
21. Leung, F., Bolloju, N.: Analyzing the Quality of Domain Models Developed by Novice Systems Analysts. In: Hawaii International Conference on Systems Sciences (2005)
22. Jourdan, L.F., Bagwell, J.J., et al.: Motivational Orientation, Self-Regulated Learning Strategies and Students' Choice of Teaching Model. Academy of Educational Leadership Journal 8(1), 59–71 (2004)
23. McElroy, J.C., Hendrickson, A.R., et al.: Dispositional Factor in Internet Use: Personality Versus Cognitive Style. MIS Quarterly 31(4), 809–820 (2007)
24. Marakas, G.M., Yi, M.Y., Johnson, R.: The multilevel and multifaceted character of computer self-efficacy: Toward a clarification of the construct and an integrative framework for research. Information Systems Research 9(2), 126–163 (1998)
25. Mason, R.O., Mitroff, I.I.: Challenging Strategic Planning Assumptions: Theory, cases, and techniques. Wiley, New York (1981)
26. Moody, M.: Best Friends. Riverhead Books (2002)
27. Lindland, O., Sindre, G., Solvberg, A.: Understanding quality in conceptual modeling. IEEE Software 11(2), 42–49 (1994)
28. Pajares, F., Graham, L.: Self-Efficacy, Motivation Constructs, and Mathematics Performance of Entering Middle School Students. Contemporary Educational Psychology 24, 124–139 (1999)
29. Rozell, E.J., Gardner, W.L.: Cognitive, motivation, and affective processes associated with computer-related performance: a path-analysis. Computers in Human Behavior 16, 199–222 (2000)
30. Ryan, S., Bordoloi, B., et al.: Acquiring Conceptual Data Modeling Skills: The Effect of Cooperative Learning and Self -Efficacy on Learning Outcomes. The Database for Advances in Information Systems 43(4), 9–24 (2000)
31. Shaft, T.M., Vessey, I.: The Role of Cognitive Fit in the Relationship Between Software Comprehension and Modification. MIS Quarterly 30(1), 29–55 (2006)
32. Simsion, G.: Data Modeling Theory and Practice. Technics Publications, LLC, Bradley Beach (2007)
33. Tiwana, A., McLean, E.R.: Expertise Integration and Creativity in Information Systems Development. Journal of Management Information Systems 22(1), 13–43 (2005)
34. Topi, H., Ramesh, V.: Human Factors Research on Data Modeling: A Review of Prior Research, An Extended Framework and Future Research Directions. Journal of Database Management 13(2), 3–19 (2002)
35. Vera-Munoz, S.C., William, J., Kinney, R., et al.: The Effects of Domain Experience and Task Presentation Format on Accountants' Information Relevance Assurance. The Accounting Review 76(3), 405–429 (2001)
36. Vessey, I.: The Effect of the Application Domain in IS Problem Solving: A Theoretical Analysis (2002)
37. Wand, Y., Weber, R.: Research Commentary: Information Systems and Conceptual Modeling - A Research Agenda. Information Systems Research 13(4), 363–376 (2002)
38. Wheeler, P.R., Hunton, J.E., et al.: Accounting Information Systems Research Opportunities Using Personality Type Theory and the Myers-Briggs Type Indicatoe. Journal of Information Systems 18(1), 1–19 (2004)

Enriching Conceptual Modelling Practices through Design Science

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. Models, modelling languages, modelling frameworks and their background have dominated conceptual modelling research and information systems engineering for last four decades. Conceptual models are mediators between the application world and the implementation or system world. Design science distinguishes the relevance cycle as the iterative process that re-inspects the application and the model, the design cycle as the iterative model development process, and the rigor cycle that aims in grounding and adding concepts developed to the knowledge base. This separation of concern into requirements engineering, model development and conceptualisation is the starting point for this paper.

Research in design science and on conceptual modelling resulted in a large body of knowledge, practices, and techniques. The two research approaches have developed their approaches and solutions. This paper shows how the two approaches can be integrated without making a sacrifice for integration. Modelling is based on modelling activities. Integration therefore starts with an integrated view on modelling. As an example of this integration we shall use reasoning support for modelling. Each modelling step considers specific work products, orients towards specific aspects of the system or application, involves different partners, and uses a variety of resources.

Keywords: conceptual modelling, design science, intellectual support for modelling, modelling processes and workflows, description, prescription, documentation, conceptualisation, explanation.

1 Introduction

1.1 Conceptual Modelling as a Complex and Multi-facetted Intellectual Process

Conceptual modelling is one of the main activities during information system development. Models are used as mediating artifacts which describe the problem to be solved for the application and which are used as starting point for implementation. They are also used for documentation of the system, for migration and evolution processes, for

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 497–510, 2011.

optimisation of systems, for control of parts of systems, and for simulation of systems. Models must reflect the structure of a system, the functionality of a system, the support facilities of a system and the collaboration environment of a system. Therefore, models are rather complex or are using multiple sub-models. Furthermore, models must be scalable to different abstractions in dependence on stakeholders such as business architects, modellers, programmers, component developers or operators.

This inherent complexity of models makes conceptual modelling itself a complex task. Conceptual modelling is thus complex and multi-facetted due to the variety of aspects to be considered and due to scaling requirements. We may use context abstraction and scoping techniques during modelling activities and thus concentrate on some of the aspects. This separation of concern supports an evolutionary approach to model development.

We also may differentiate modelling activities by problem kinds that are of importance at a current modelling step. For instance, the abstraction layer model [35] distinguishes the application domain layer, the business process layer, the business operating layer, the conceptual layer and implementation layers. With such distinction we realise that tasks at each level are of different nature. For instance, at the application domain layer we describe the way how applications are operating, which application processes must be supported by which application data at which moment of time in which format, for which reason, by and for whom, and in which environments.

1.2 Design Science versus or Joined with Conceptual Modelling

Design science started with an analysis and assessment of research activities that are used for system development in computer engineering. The separation into the application world, the modelling and model world and the knowledge or science world [13,39] supported an assessment of the results of modelling and an evaluation of the results of research on modelling. The application world is characterised by three dimensions: people, organisations and the (technical) infrastructure. The science world is based on scientific theories with specific paradigms and justification underpinnings from one side and techniques for deployment of these theories on the other side. The modelling and model world is supported by the science world from one side and governed by the application world. This mediating role of models and modelling results clarifies the importance, value and application of models.

Conceptual modelling has been oriented in the past mainly to clarification on languages, on methods for deployment of such languages, on (mathematical) theories as foundations of syntactics, semantics and pragmatics of model, and on evaluation and quality guaranteeing methods [27,37,39]. The application world is used as a starting point for the development of systems that solve some problems of the application domain under consideration. The specific objectives of the application domain are mapped to requirements which are then used as the main governing background for the system development. A theory of conceptual models and of conceptual modelling is currently under development [38,39].

If we analyse these two directions we come to a conclusion similar to [45]. In reality design science research and research on conceptual modelling are only two sides of the same coin. The two communities behind the two sides of the coin are currently engaged

in a discussion of the added value of each side. We take another direction in this paper. We will show how these two sides can be neatly integrated. We realise that it is natural to integrate conceptual modelling theories and design science. We target at a revision and extension of classical information systems modelling methodologies and at overcoming limitations of such modelling processes. This paper is based classical notions[1] which are used for IS design and construction and extends the theory of conceptual models [39] by design science research. It generalises co-design for information systems [34,35,36], design science research [13,23], business information systems engineering [11,17], and different frameworks, e.g. FRISCO [12]. Based on this approach we enrich conceptual modelling practices by design science research.

1.3 Structure of the Paper

We first revisit design science and discover how design science and classical approaches to information systems development can be integrated. Section 4 is based on the theory of conceptual models [6,39] and of modelling activities [38]. We introduce five of the main workflows of modelling and restrict consideration to IS development processes, i.e., we base development on description of IS followed by prescription for construction of IS. Section 3 develops a path to integration of design science and conceptual modelling. Section 4 uses this approach for homogenisation of activities that are both used in design science and conceptual modelling. We show that design science and conceptual modelling theory can be neatly integrated.

2 Design Science Revisited

Information systems research (ISR) lies at the intersection of people, organizations, and technology [31]. It relies on and contributes to cognitive science, organizational theory, management sciences, and computer science. It is both an organizational and a technical discipline that is concerned with the analysis, construction, deployment, use, evaluation, evolution, and management of information system artifacts in organizational settings [21,28,40]. Within this setting, the design-science research paradigm is proactive with respect to technology. It focuses on creating and evaluating innovative IT artifacts that enable organizations to address important information-related tasks.

2.1 Design Research in Information Systems

The genesis of design science lies in Herbert Simon's The Sciences of the Artificial (first published in 1969) in which he articulated the difference between natural science, concerned with how things are, and design science, concerned with how things ought to be, based on his understanding of design as problem solving [32]. Following Simon's tradition, design science was introduced in IS (information systems) research by March and Smith [22], who presented it as prescriptive research aimed at improving

[1] See, for instance, chapters in the collection [6, chapters 1 (Modelling as programming), 2 (MMD), 6 (Extended ER models), 7 (Codesign), 8 (Business processes), 10 (Interaction models), 11 (Web IS development), 12 (Evolution and migration), 13 (Data integration)].

ICT (information, communication, and technology) performance, as opposed to natural science, corresponding to descriptive research aimed at understanding the nature of ICT. An important point was that IS research should actually integrate both perspectives, an argument that came back on Hevner, et. al., [13], establishing design science research in information systems (DSRIS). A DSRIS contribution requires identifying a relevant organizational ICT problem, demonstrating that no solution exists, developing an ICT artifact that addresses this problem, rigorously evaluating the artifact, articulating the contribution to the ICT knowledge-base and to practice, and explaining the implications for ICT management and practice [23].

2.2 The Three Design Science Research Cycles

A recent development of the initial design science framework presented in [13], decouples the previous goals into three distinct but interrelated research cycles, as shown in Figure 1. This three-cycle view of design science suggests that relevance is attained through identification of requirements (or business needs) and field testing of an artifact within an environment, while rigor is achieved by appropriately grounding the research in existing foundations, methodologies and design theories and subsequently making contributions that add to the existing knowledge base. The design aspect is achieved through a *design cycle* in which the artifact must be built and evaluated thoroughly before "releasing it" to the *relevance cycle* and before the knowledge contribution is output into the *rigor cycle* [14].

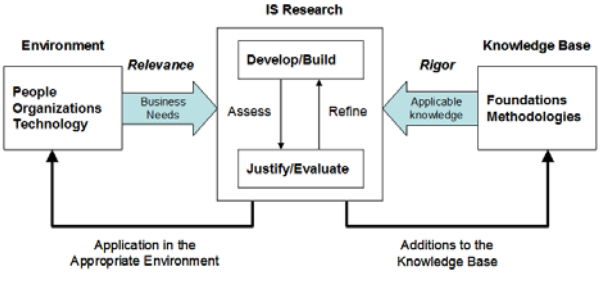


Fig. 1. Design science research cycles as introduced by A.R. Hevner et al. [13]

2.3 Strengths, Weaknesses, Opportunities and Threats of Design Science Research

IS design is successful when it has a contribution that applies a range of methods, when is guided by vision, and when it has a positive impact on society. Design science has relevance in all academic systems even though rarely cited in other fields [9]. Design science has achieved a hegemonic status and has become an orthodoxy instead of constantly evolving set of precepts taking its legitimate place amongst the panoply of concerns the IS community needs to deal with [19]. Those concerns characterized design science as a "wicked" problem that needs 'rethinking' in order to elevate DSRIS to the equal footing with other research paradigms in IS research [2, 16, 29, 20].

The dual goal of DSRIS is producing both an artifact and a theoretical contribution. March and Smith [22] attach the activities of Discovery (generating or proposing scientific claims) and Justification (testing scientific claims for validity) to natural science and present them as separate from (but parallel to) the activities of Building (constructing an artifact for a specific purpose) and Evaluation (determining how well the artifact performs) attached to design science. In Hevner et al. [13], the activities were merged into Develop/Build and Justify/Evaluate. This helps state the case in favor of having both relevance and rigor in ISR, but leave behind lack of clarity with regards to how theory development should be seen in DSRIS.

On one end of the spectrum, March and Smith [13] explicitly exclude theory and theorizing from design science. On the other end, several authors contend that theory development should be an integral part of DSRIS [18,24,43]. Hevner et al. [13] do not seem to take a stance either way, since they present a dual build/develop and evaluate/justify design cycle, potentially allowing for both artifact building and evaluation as well as theory development and justification.

When DSRIS is used for theory development, the next question is what kind of theory can result. Walls et al. [43] speak of design theories, which are prescriptive theories about how to design information systems effectively and feasibly. Venable [42] claims that design theories should be reduced to utility theories, which are predictive (rather than prescriptive) about the utility of applying a meta-design to solve meta-requirements. Theory can also be related to the kinds of artifacts produced by DSRIS, which according to [13] may be constructs, models, methods and/or instantiations. For Winter [44] theories should be considered a fifth (intermediate) type of artifact. In contrast, Gregor and Jones [9] take a "broad view of theory" which encompasses constructs, models and methods, and where only instantiations correspond to the (material) artifact as such. Theory in this last sense is equivalent to what is termed elsewhere conjectures, models, frameworks, or bodies of knowledge, so the three outputs of design science, besides the instantiated artifact, are regarded as "components of theory" [44]. The options for a theory being (a) prescriptive design theory, (b) predictive utility theory, (c) an additional intermediate artifact or (d) all abstract artifacts.

According to Iivari [15], the different types of knowledge produced are what determine the epistemology underlying DSRIS. Gregor [8], however, has strongly argued that the type of theory produced by ISR should not depend on the underlying paradigm; for her, theory is independent from specific ontological or epistemological choices. Nonetheless, both recognize the importance of making explicit choices about the philosophy underlying the research. Although DSRIS is not specific in terms of an underlying epistemology, it may be seen as rooted in pragmatism in the sense that it emphasizes utility (the measure of truth in pragmatism) [13]. This assumption, although widely held, does not hold for all kinds of DSRIS, making it open to alternative epistemology, such as interpretivism [15,7,26].

A check list for understanding design science is Baskerville's [1] what design science is not list: Design science is not design, is not design theory, is not an IT artifact, is not a methodology, is not action research, is not computer science, is not a separate academic discipline, and it is not new. Also, design science research is in an up-hill struggle in defining the term design science [1,33].

First, in order to show the path for integrated understanding we harmonise Hevner et al. [13] and Winter [44] interpretations of design science research in information systems as the combination of two different types of information systems research contributions: (IS) design science as contributions that reflect at a generic level the construction and evaluation of artifacts and (IS) design research as construction and evaluation of specific artifacts. (IS) design science research agrees on the differentiation of constructs, models, methods and instantiations as types of design science research artifacts [3,22,40,44].

Second, we correlate notions used in design science research and information systems design and construction in Figure 2 between design science research and (professional) information systems design and construction.

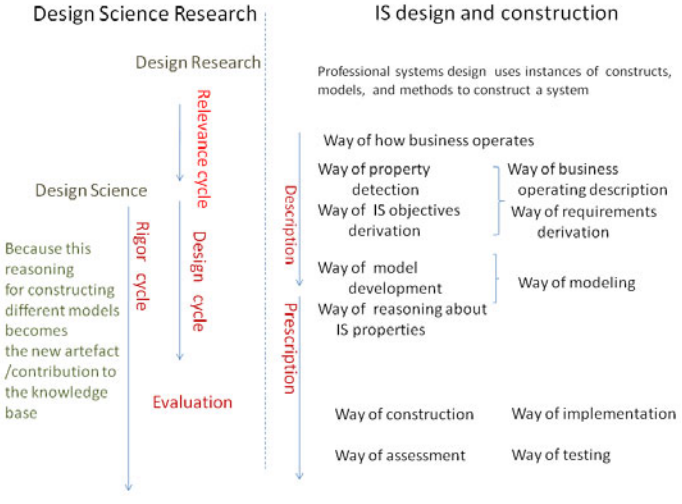


Fig. 2. Framework for understanding Design Science Research and Information Systems Design & Construction

The main difference between (professional) information systems design and construction and design science research is that; design science research (DSR) contribution requires identifying a relevant organizational ICT problem, demonstrating that no solution exists, developing an ICT artifact that addresses this problem, rigorously evaluating the artifact, articulating the contribution to the ICT knowledge-base and to practice, and explaining the implications for ICT management and practice [23]. The professional system design and construction use the available artifacts.

The main drawback in conducting design science research is the lack of a theory for construction of artifacts that supports the design cycle. Therefore, design science research needs to explore the types of models and reasoning behind model construction (actually this is conceptual modelling theory). The rigor cycle in design science targets at creation of design/modelling knowledge. It considers the the model or artifact constructed. The relevance and the design cycles of design research target at construction of artifacts. These two facets of DSR require a deep integration and sophisticated reasoning support.

2.4 Alternatives for a Notion of a Theory

The notion of a theory itself is be a matter of intensive research [5,25]. We base our understanding of the notion of a theory on the three dimensions and the main goal of conceptual modelling.The classical treatment of the notion of a theory is based on mathematical and philosophical logics and is far too strict. We may however inherit certain elements of such logics. Already the notion of semantics provides a larger number of choices [30] beyond those that are taken for granted in Computer Science.

Theories of conceptual modelling must step beyond axiomatic and analytical theories. They must also be operational and 'genetic'. Theories of conceptual modelling can be developed in the frameworks of logical empiricism, of context theories ('context of use', 'language game'), and of constructivism. The first framework supports to define purposes of conceptual modelling, to emphasise threats that should be handled with the help of models, to select appropriate modelling languages and methods, to reason on the quality of the model depending on the purpose of the model, to select measures for the quality of models, and to guide the process of modelling. The second framework relates models to the application domain, to the stakeholders participating in the development process, to the aspects reflected within a model, and to the resources provided either by the system and by the knowledge from the application domain. The last framework provides a basis for a general structure by a *language of constructs* that can be applied for the development of a model, a *set of constructors* that can be applied to combine models into a new model, and a *number of quality properties for characterisation of usage* of certain constructs.

3 Integration of Design Science and Conceptual Modelling

3.1 Associating Design Science Approaches with Conceptual Modelling

Conceptual modelling within the business information community and design science approaches are two sides of the same 'Janus' head of a coin. IS design science aims at the development of a general theory for models, modelling activities and modelling. We shall use the approach for a deeper insight into modelling. We claim that each enhance the other. We must however integrate the vocabulary used in the two approaches. For instance, models are called 'design' in [13] and 'theories' in [9].

The information systems community characterises the modelling process by seven guidelines [13]:

(1) model are purposeful IT artifacts created to address a problem;
(2) models are solutions to relevant and important problems;
(3) the utility, quality, and efficacy of models must be evaluated by quality assessment;
(4) modelling research must contribute to the state of the art;
(5) modelling research relies upon the application of rigorous methods;
(6) modelling is a search process and use termination conditions;
(7) models must be communicated both to technology-oriented as well as to management audiences.

We observe that guidelines (1), (2), and (7) are characterising the model. Guidelines (3), (6) characterise modelling activities. Guidelines (3) and (5) are related to modelling as

a technology. Guideline (4) is a general statement that raise the issue whether modelling may become a science.

Main ingredients of modelling can be derived from these guidelines [9,23]. Core components are purpose and scope (causa finalis), artifacts (causa materialis), the oneness of form and function (causa formalis), artifacts mutability, testable propositions about the model, and theoretical underpinning. Additional requests are the potential implementation (causa efficiens) and utility for exposition and testing [9].

3.2 Associating Design Science Cycles with Main Activities of IS Modelling

Design science separates three cycles [41]: the relevance (or description) cycle, the design (or modelling) cycle, and the rigor (or conceptualisation and generalisation or knowledge development) cycle.

The rigor cycle also aims at the development of knowledge about the application domain and the model. This part of the rigor cycle is *conceptualisation*. The second target of the rigor cycle is the derivation of abstract knowledge and experience, of scientific theories that can be applied in similar application cases, of (pragmatical) experience for modelling, and of meta-artifact or reference models based on model-driven development (MDD) approaches. Design science aims at another kind of model refinement by adding more rigor after evaluation of a model. This refinement is essentially *model evolution* and *model evaluation*. Another refinement is the enhancement of models by concepts. This refinement is essentially a 'semantification' or *conceptualisation* of the model.

We observe that the rigor cycle or stage is orthogonal to the modelling and relevance cycles. The modelling cycle may be broken into a description stage that relates the application domain to the model and a prescription stage that uses the model for system construction. The rigor cycle or stage is somehow orthogonal. It has at least two facets: one facet that is important for the model and one facet that is important for generalisation of the model, e.g., for derivation of pattern or reference models and for extraction of model and modelling knowledge beyond the actual modelling activity. In the sequel we concentrate on the first facet of the rigor cycle.

3.3 Stages Observed in Conceptual Modelling

We can now summarise our revision of design science. Design science uses the three cycles: relevance, modelling, and rigor cycle. The main purpose of conceptual modelling of information systems is the construction of an information system. This construction is based on description of the application domain and the of the model. We reflect some relevant properties of the application domain by obligations to the model and by the model. This model is typically used as a prescription for the construction. Therefore we distinguish the stages displayed in Figure 3.

The relevance cycle used in design science contains the modelling stage and a small fraction of the realisation stage. Conceptual modelling is not explicitly considering the relevance stage. It entirely uses the modelling and the realisation stage. Conceptualisation is an orthogonal activity that aims at a theoretical underpinning of models. It is used

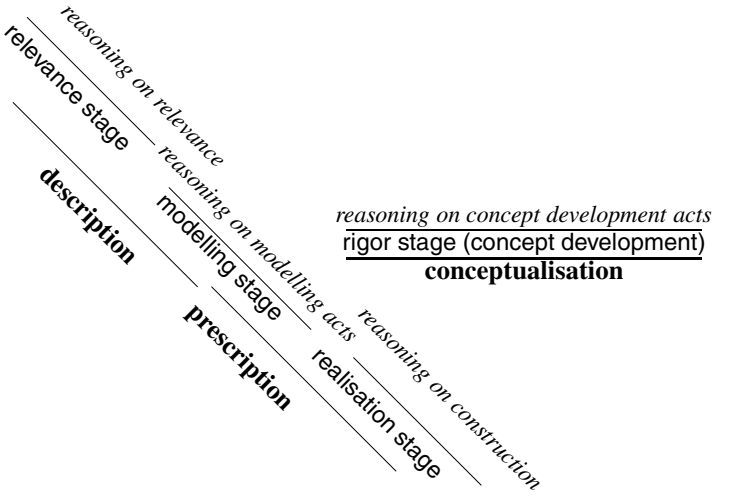


Fig. 3. The stages of conceptual modelling activities for the description-prescription workflow enhanced by conceptualisation

for *semantification* of models and for improvement of comprehensibility of models and elements used in models.

4 An Integration of Modelling Activities

Design science modelling and classical conceptual modelling of information systems mainly aim at construction of systems. Therefore, we may also restrict this paper to activities that support construction. In the sequel we introduce some of the main macro-activities and derive corresponding workflows: description, prescription, conceptualisation, explanation, and documentation. Other macro-activities are: exploration; optimisation and variation; verification.

Modelling is typically supported by languages that are well-founded and easy to apply for the description of the application domain, the requirements and the system solution. Modelling activities are ruled by the *purpose* of the model. The purpose in the center of this paper is construction of an artifact or of an information system. Beside this purpose there are many other purposes such as perception support for understanding the application domain, explanation and demonstration for understanding an origin, preparation to management and handling of the origin, optimisation of the origin, hypothesis verification through the model, control of parts of the application, simulation of behaviour in certain situations, and substitution for a part of the application.

The construction purpose is supported by a consideration of (1) the part of the application domain (called origin) that is of interest, (2) the model that is used for construction and that reflect the origin, (3) the enhancement of the model by concepts for conceptualisation, documentation and explanation, and (4) the realisation facilities used for the construction of the information system which is a technical artifact.

4.1 Description of Origin and Reflection by the Model

The application domain consists of people, organisational systems, and technical systems that interact to work towards a goal. This description of the application domain clarifies problems to be solved depending on models purpose, properties reflected or neglected, the scope to specific elements, the restrictions of the world associated with the model.

A *model* is typically a schematic description of a system, theory, or phenomenon of an origin that accounts for known or inferred properties of the origin and may be used for further study of characteristics of the origin.

The description of the origin depends on the language $\mathcal{L}_{AD}$ used within the application domain and on the theories $Theor(O)$ behind the origin. It results in a number of properties $\Phi(O)$ of the origin. These properties must be understood and mapped to objectives $\Psi(A)$ for the model A. This mapping depends on the language $\mathcal{L}_{artifacts}$ for artifacts used for the model. Figure 4 represents the association between an origin and a model.

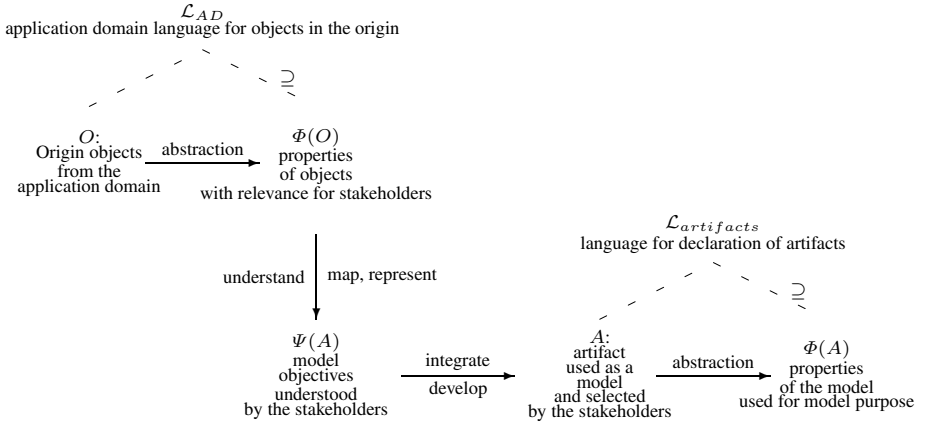


Fig. 4. Modelling for description of the origin and as the basis of realisation by a technical artifact

4.2 Prescription for Construction of Systems

Properties $\Phi(A)$ of the model A rule the construction of a system in the realisation stage. Properties also incorporate static and dynamic integrity constraints [35]. Typically we assume that the model and the properties are combined within a specification. Modern methodologies are based on interactive automatic mappings to the technical artifact TA. These properties serve as objectives for the technical artifact development, i.e. *prescribing* the information system.

4.3 Conceptualisation of Models

Conceptualisation extends the model by a number of concepts that are the basis for an understanding of the model and for the explanation of the model to the user. [39] introduces a general theory of concepts that can be used for conceptualisation. Concepts

are used in everyday life as a communication vehicle and as a reasoning chunk. Concept definition can be given in a narrative informal form, in a formal way, by reference to some other definitions etc. We may use a large variety of semantics [30], e.g., lexical or ontological, logical, or reflective.

Our revision of the design science and the conceptual modelling frameworks in Figure 3 shows that concept enhancement is an orthogonal activity. It can be partially completed without having a negative impact on the realisation phase if stakeholders involved have a common understanding of the model and its properties and a commonsense about the objectives for realisation and a good mapping facility. Therefore, conceptualisation may also be implicit and use some kind of lexical semantics, e.g. word semantics within a commonly agreed name space.

4.4 Explanation

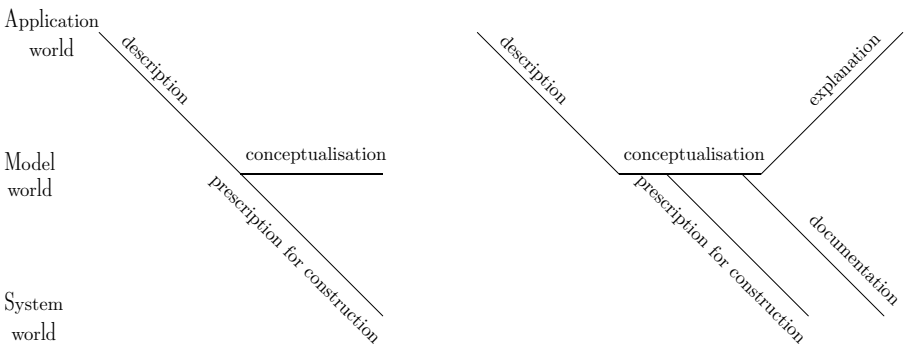
According to [10], MIS distinguishes four main tasks: description of application domains, construction of systems, prognosis of the impact of the IS, and finally explanation to business users. Explanation is often understood as a self-description of the model and thus mistreated in discussions [29,16]. Design science research and behaviouristic approaches use explanation for reasoning and/or observing on system or model properties in terms of the application world. Reasoning is thus based on deductive-nomological methods and methods of empirical research.

4.5 Documentation Based on Modelling Results

Classical information systems development follows sometimes the late documentation approach, i.e. the development documentation is based on the final result modelling and/or realisation and partial consideration of the conceptualisation, i.e. consists of the model and its translation to relational or object-relational system languages. Documentation should however reflect all thoughts and decisions applied in modelling and realisation, i.e., record all reasoning, construction and decision processes for model and system development.

4.6 Modelling Workflows

So far we introduced different stages such as the relevance stage, the modelling stage, and the realisation stage. These stages can be combined with conceptualisation, explanation and documentation. Figure 5 shows how these stages can be combined. Often, system development is based on agile development, i.e., we use description followed by prescription. This development process may be enhanced by adding more rigor to the model, i.e., concept enhancement of the model. This rigor is also partially useful for the system construction. Therefore, we arrive at the picture at the right side of Figure 5. Additional conceptualisation might be used for system development documentation. The full conceptualisation is a good starting point for an explanation of the system to the real life business users and for empirical studies in the application domain.



(a) Direct realisation without considering concepts

(b) Realisation with consideration of concepts

Fig. 5. Two approaches to systems development

5 Conclusion

Based on the definition of two types of design science research contributions (design science and design research) we illustrated the parallels between design science research and professional information systems design and construction. Mapping the relevance and design cycle to professional information systems modelling and models, the overlaps and gaps within the descriptive and prescriptive modelling needs were identified and presented. This gap analysis led to the integration of reasoning in design science and conceptual modelling and is a source of inspiration for abstraction and theory formation. Our research demonstrates the potential of an integration of design science research from one side and of conceptual modelling from the other side.

Even though, presented and discussed separately, research in design science and on conceptual modelling resulted in a large body of knowledge, practices, and techniques, contributing to each other. We have presented a starting point that can generate many contributions leading to fruitful discussions on the inter relationships and contributions of design science research and conceptual modelling. This paper is the first series of articles that will devote to improve the understanding, parallels, similarities and contributions of to each other in design science research and conceptual modelling.

Our revision and integration of design science and conceptual modelling frameworks led to a better understanding of potential relationship of conceptual modelling and design science. We have been using an integrated model. Alternatively we could use a model suite [4] consisting of combined models, e.g., a description model, a prescription model, a conceptual model, an explanation model, a documentation, etc.

References

1. Baskerville, R.: What design science is not. EJIS 17(5), 441–443 (2008)
2. Baskerville, R., Lyytinen, K., Sambamurthy, V., Straub, D.: A response to the design-oriented information systems research memorandum. European Journal of Information Systems, 1–5 (2010)

3. Vom Brocke, J., Buddendick, C.: Reusable conceptual models - requirements based on the design science research paradigm. In: DESRIST, pp. 576–604 (2006)
4. Dahanayake, A., Thalheim, B.: Co-evolution of (Information) system models. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) BPMDS 2010 and EMMSAD 2010. Lecture Notes in Business Information Processing, vol. 50, pp. 314–326. Springer, Heidelberg (2010)
5. Deppert, W.: Theorie der Wissenschaft. Christian-Albrechts-University at Kiel, Lecture notes for academic year 2008/2009 (2009), http://wolfgang.deppert.de
6. Embley, D., Thalheim, B. (eds.): Handbook of conceptual modelling: Its Usage and Its Challenges. Springer, Heidelberg (2011)
7. Klabbers, J.H.G.: A framework for artifact assessment and theory testing. Simulation & Gaming 37(2), 155–173 (2006)
8. Gregor, S.: The nature of theory in information systems. MIS Quarterly 30(3), 611–642 (2006)
9. Gregor, S., Jones, D.: The anatomy of a design theory. Journal of Association for Information Systems 8(5), 312–335 (2007)
10. Heinrich, L.J., Heinzl, A., Riedl, R.: Wirtschaftsinformatik: Einführung und Grundlegung, 4th edn. Springer, Berlin (2011)
11. Hesse, W.: Dinosaur meets Archaeopteryx? or: Is there an alternative for Rational's Unified Process? Software and System Modeling 2(4), 240–247 (2003)
12. Hesse, W., Verrijn-Stuart, A.A.: Towards a theory of information systems: The FRISCO approach. In: EJC, pp. 81–91 (2000)
13. Hevner, A., March, S., Park, J., Ram, S.: Design science in information systems research. MIS Quaterly 28(1), 75–105 (2004)
14. Hevner, A.R.: The three cycle view of design science. Scandinavian J. Inf. Systems 19(2) (2007)
15. Iivari, J.: A paradigmatic analysis of information systems as a design science. Scandinavian J. Inf. Systems 19(2) (2007)
16. Junglas, I., Niehaves, B., Spiekermann, S., Stahl, B.C., Weitzel, T., Winter, R., Baskerville, R.: The inflation of academic intellectual capital: the case for design science research in europe. European Journal of Information Systems, 1–6 (2010)
17. Koch, S., Strecker, S., Frank, U.: Conceptual modelling as a new entry in the bazaar: The open model approach. In: OSS. IFIP, vol. 203, pp. 9–20. Springer, Heidelberg (2006)
18. Kuechler, B., Vaishnavi, V.: On theory development in design science research: Anatomy of a research project. European Journal of Information Systems 17(5), 489–504 (2008)
19. Land, F., Loebbecke, C., Angehrn, A.A., Clemons, E.K., Hevner, A.R., Mueller, G.: ICSI 2008 panel report: Design science in information systems: Hegemony, bandwagon, or new wave? Communication of the Association for Information Systems 24(29), 501–508 (2009)
20. Lyytinen, K., Baskerville, R., Iivari, J., Téeni, D.: Why the old world cannot publish? overcoming challenges in publishing high-impact is research. EJIS 16(4), 317–326 (2007)
21. Madnick, S.E.: The challenge: To be part of the solution instead of being part of the problem. In: Second Annual Workshop on Information Technology and Systems, pp. 1–9 (1992)
22. March, S.T., Smith, G.F.: Design and natural science research on information technology. Decision Support Systems 15(4), 251–266 (1995)
23. March, S.T., Storey, V.C.: Design science in the information systems discipline: An introduction to the special issue on design science research. MIS Quarterly 4, 725–730 (2008)
24. Markus, M.L., Majchrzak, A., Gasser, L.: A design theory for systems that support emergent knowledge processes. MIS Quarterly 26(3), 179–212 (2002)
25. Mittelstraß, J. (ed.): Enzyklopädie Philosophie und Wissenschaftstheorie. J.B. Metzler, Stuttgart (2004)

26. Niehaves, B.: On epistemological pluralism in design science. Scandinavian J. Inf. Systems 19(2) (2007)
27. Olivé, A.: Conceptual modeling of information systems. Springer, Berlin (2007)
28. Orlikowski, W.J.: Using technology and constituting structures: A practice lens for studying technology in organizations. Organization Science 11(4), 404–428 (2000)
29. Österle, H., Becker, J., Frank, U., Hess, T., Karagiannis, D., Krcmar, H., Loos, P., Mertens, P., Oberweis, A., Sinz, E.J.: Memorandum on design-oriented information systems research. European Journal of Information Systems, 1–4 (2010)
30. Schewe, K.-D., Thalheim, B.: Semantics in data and knowledge bases. In: Schewe, K.-D., Thalheim, B. (eds.) SDKB 2008. LNCS, vol. 4925, pp. 1–25. Springer, Heidelberg (2008)
31. Silver, M.S., Markus, M.L., Beath, C.M.: The information technology interaction model: A foundation for the MBA core course. MIS Quarterly 19(3), 361–390 (1995)
32. Simon, H.: The Sciences of the Artificial. MIT Press, Cambridge (1981)
33. Sutton, R.I., Staw, B.M.: What theory is not. Administrative Science Quartly 40(3), 371–384 (1995)
34. Thalheim, B.: Codesign of structures, functions and interfaces in database applications. Technical Report Preprint I-05-1997, Brandenburg University of Technology at Cottbus, Institute of Computer Science (1997) (in German)
35. Thalheim, B.: Entity-relationship modeling – Foundations of database technology. Springer, Berlin (2000)
36. Thalheim, B.: Co-design of structuring, funtionality, distribution, and interactivity of large information systems. Computer Science Reports 15/03, Cottbus University of Technology, Computer Science Institute (2003)
37. Thalheim, B.: Database component ware. In: ADC 2003. Australian Computer Science Communications, vol. 25(2), pp. 13–26 (2003)
38. Thalheim, B.: Towards a theory of conceptual modelling. Journal of Universal Computer Science 16(20), 3102–3137 (2010), http://www.jucs.org/jucs_16_20/towards_a_theory_of
39. Thalheim, B.: The theory of conceptual models, the theory of conceptual modelling and foundations of conceptual modelling. In: The Handbook of Conceptual Modeling: Its Usage and Its Challenges, ch. 17, pp. 547–580. Springer, Berlin (2011)
40. Vahidov, R.: Design researcher's is artifact: a representational framework. In: DESRIST, pp. 9–33 (2006)
41. Venable, J.R.: Design science research post hevner et al.: Criteria, standards, guidelines, and expectations. In: Winter, R., Zhao, J.L., Aier, S. (eds.) DESRIST 2010. LNCS, vol. 6105, pp. 109–123. Springer, Heidelberg (2010)
42. Venable, J.R.: The role of theory and theorising in design science research. In: DESRIST (2006)
43. Walls, J.G., Widmeyer, G.R., El Sawy, O.: Building an information system design theory for vigilant eis. Information Systems Research 3(1), 36–59 (1992)
44. Winter, R.: Design science research in Europe. European Journal of Information Systems 17(5), 470–475 (2008)
45. Zelewski, S.: Kann Wissenschaftstheorie behilflich für die Publikationspraxis sein? In: Lehner, F., Zelewski, S. (eds.) Wissenschaftstheoretische Fundierung und wissenschaftliche Orientierung der Wirtschaftsinformatik, pp. 71–120. GTO (2007)

A Meta-language for Enterprise Architecture Analysis

Sabine Buckl[1], Markus Buschle[2], Pontus Johnson[2],
Florian Matthes[1], and Christian M. Schweda[1]

[1] Chair for Software Engineering of Business Information Systems (sebis),
Technische Universität München,
Boltzmannster. 3, 85748 Garching, Germany
{sabine.buckl,matthes,christian.m.schweda}@mytum.de
[2] Industrial Information and Control Systems,
KTH Royal Institute of Technology,
Osquldas v. 12, SE-10044 Stockholm, Sweden
{markusb,pontus}@ics.kth.se

Abstract. Enterprise Architecture (EA) management is a commonly accepted instrument to support strategic decision making. The objective of EA management is to improve business IT alignment by making the impact of planned changes explicit. The increasing interconnectivity of applications with other applications and with business processes however makes it difficult to get a complete view on change impacts and dependency structures. This information is nevertheless required to support decision makers. Current meta-languages proposed for the context of EA management provide only limited support for modelling qualitative and quantitative dependencies.

In this paper we propose a meta-language, which builds on the Meta Object Facility (MOF). This meta-language specifically accounts for the requirements of EA analysis. We discuss existing meta-languages from the field of EA management and related areas against these requirements. Building on the standard of the OMG, we present an extension of MOF designed to support EA analysis. The theoretic exposition of the extension is complemented by an example illustrating the applicability of the presented meta-language.

1 Introduction and Motivation

Today, the strategic management of the Enterprise Architecture (EA) is a commonly accepted instrument of modern enterprises. EA is used to keep up with the increasing demand for flexible IT support and the overall managed evolution of the enterprise. Therein EA analysis [1] is a means of providing decision support throughout the management process by making the impact of planned projects explicit. In the complex and interwoven system "enterprise", local changes to one artifact, e.g. a business process or a business application, might have unforeseen global consequences and potentially detrimental impacts on related artifacts.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 511–525, 2011.

Therefore, it is difficult to get a complete view on change impacts and dependency structures, this is nevertheless required to support decision making.

With regards to the increased interest from practitioners, different approaches to EA management have been developed in academic research [2,3,4], by practitioners [5,6], standardization bodies [7,8], and tool vendors [9]. These approaches provide frameworks, methods, and models used in the design of an EA management function. Thereby, the models typically focus on *structural* aspects of the EA, such as dependencies between business processes and business applications. Decisions about the future of the EA are made based on the analysis of these dependencies and plans for a managed evolution are created (cf. [10]).

Input to the aforementioned analysis are (parts of the) architectural descriptions of the EA. A variety of modelling techniques and complementing information models underlying these architectural descriptions exists. We understand an *information model* in line with Buckl et al. in [11] as "a model which specifies, which information about the ea, its elements and their relationships should be documented, and how the respective information should be structured". Existing information models differ widely with respect to the concepts that they employ as well as the coverage of the EA that they aim at. This diversity backs the assumption of researchers in the EA management domain, that information models represent organization-specific artifacts (cf. [11,12]).

The models described above have in common that they are implicitly or explicitly based on meta-languages. In the context of EA management a widely used meta-language for EA information modelling is the general purpose modelling language *unified modelling language* (UML) [13]. UML and other meta-languages proposed for the context of EA management provide only limited support for modelling quantitative and qualitative dependencies. In particular the latter are of interest in the context of EA management. They express that *some kind* of dependency exists between attributes, while not having to indicate of *what kind* this dependency is. The aforementioned problem statement that we seek to address can be summarized as follows:

> How does a meta-language look like that supports EA analysis by enabling the user to model different types of dependencies and attributes?

In this paper, we propose a meta-language that builds on the *Meta Object Facility* (MOF) [14] and specifically accounts for the requirements of EA analysis. This approach was chosen as MOF represents the basis of UML a frequently used meta-language for the context of EA management [15]. We prepare the exposition of our solution by eliciting requirements that such a meta-language should fulfill in Section 2. Based on the requirements we revisit related work from the EA management domain and related fields in Section 3. In Section 4 we discuss an extension of *essential* MOF (EMOF) as a meta-language for EA analysis. Our meta-language addresses the afore elicited requirements by providing means for specifying, that the values of certain attributes are dependent on other attributes' values without creating the need to provide computable dependency rules. The theoretic exposition of our meta-language is complemented by

an example demonstrating applicability of the presented solution in Section 5. Finally, we conclude the paper with a discussion on further areas of research.

2 Requirements on the Meta-language

This section presents requirements on the meta-language we propose. In the previous chapter we already described the fact that the language should extend MOF. This implies that the meta-language should feature classes that can be linked via relations, and are characterisable through attributes. These characteristics are taken from MOF right away. Besides those requirements several more prerequisites need to be considered with regards to the EA domain. These needs have been derived from [16] and [17], which both explicitly state requirements on an expressive meta-language for EA analysis. Especially the characteristics of attributes are discussed in both previously mentioned publications.

It has been stated in [16] and [18] that EA models are likely to become (partly) out-dated after short time periods. The fact that a certain application might for example be phased-out and replaced, is not necessarily resulting in an immediate update of the EA model. If such a discrepancy between the status quo in the real world and the model is not eliminated it is likely that the architectural model does not any longer represent the configuration that is in use. This phenomenon has also been identified in [17] and described as empirical uncertainty.

An expressive meta-language is required to be able to capture whether a model is likely to reflect the setting it is meant to describe (**Requirement R1**). This means that a meta-language should allow to annotate each modeled class with a probability that this modeled entity still is in use (**Requirement R1.1**). To consider only the entities of a model is however not sufficient, moreover it needs to be considered whether the modeled concepts are still related to the same entities i.e. whether they still interact in the same manner with their environment as it was described. Here one could think of a modeled application that in comparison to when it was described in a model nowadays supports a billing business process instead of a order handling process. This means that the application's relation to the order handling process does not exist any longer. The implication is that not only entities but also their relations should be describable with regards to their existence (**Requirement R1.2**). Therefore both entities and their relations need to be equipped with an additional build-in *existence* attribute reflecting the probability that the concept is still employed and collaborating with the artifacts it is related to.

A second aspect of relevance that was identified targets the expressiveness of the attributes offered by the meta-language. While typical general purpose modelling languages as UML can be used to model discrete phenomena, e.g. the number of components of an application system, creating EA models is often accompanied by uncertainty about the actual value of an attribute. The build-in data-types offered by UML and thereby MOF, for example *Boolean* or *Integer*, can be used to describe properties of classes, e.g. a *Boolean* attribute can be either *true* or *false* but never in an intermediate state. This means also that

attributes in MOF are considered to be fixed i.e one is sure that a property is in a certain state. To use distributions and thereby describe uncertainty that for example the attribute availability of a system is $95 \pm 2\%$ cannot be captured. The usefulness of continuous variables going along with the ability to describe distributions within the domain of EA analysis has been exemplified in [19] and [20]. In both publications distributions are used to describe modeled entities. These distributions then were used as input to perform EA analysis. In [19] the usage of *parametric dependencies* for performance prediction is suggested. On the other hand Närman et. al. [20] notes that reliability can be captured through the usage of continuous attributes. In order to support this kind of evaluation a power-full EA meta-language based on MOF needs to extent the attribute concept through an introduction of attributes that are capable to describe distributions and thereby express uncertainty (**Requirement R2**).

The third needed extension to MOF that was identified is the capability to express dependencies between attributes (**Requirement R3**). Attribute relationships are part of the Probabilistic Relational Models (PRM) formalism [21] (see explanation provided in 3.3), which has been proven to be a powerful mechanism to use for EA analysis. In [22] PRMs have been applied for (cyber) security analysis, whereas in [23] PRMs have been used to perform reliability evaluation. Thirdly in [24] PRMs are used to define patterns for quantitative EA analysis. The support of EA analysis using a formalism similar to PRM would facilitate the previously mentioned evaluations. This can be achieved for MOF through the introduction of relations between attributes (**Requirement R3.1**) and thus the modelling of their effects on each other (**Requirement R3.2**).

3 Related Work

In this section we discuss two modelling approaches targeting enterprises and their architectures, respectively. Thereby, we do not focus on the concepts provided by the approaches, but put emphasis on the meta-languages that underlie these approaches. We discuss if and how these meta-languages address the requirements as stated in Section 2. In addition to these approaches, we also explain probabilistic relationships models as a promising means to incorporate uncertainty in domain models and to express relations between attributes.

3.1 ArchiMate (ORM)

The ArchiMate modelling language has a long history as an approach to model EAs. In the early work [25], the approach was based on an informally defined meta-language consisting of "things" and "relationships". In the most recent publications [26] this meta-language is replaced with the *Object-Role Model* (ORM) [27]. ORM supports basic modelling concepts grounded in the dichotomy between instance and corresponding entity type. Entity types can participate in n-ary relationships with each other and with value types, which are used to designate properties in an object-oriented sense. Additional facilities in ORM enable

to model interrelationships between different relationships, e.g. describing that the set of values of one relationship, i.e. its tuples, is a subset of another relationship's values. In [4] Lankhorst et al. describe how architectural models based on ORM can be augmented with dependency information used for quantitative analyses of EAs. The modelling mechanisms are applied exemplarily, calling for an intuitive understanding of the underlying concepts. These concepts, which would contribute an extension to ORM, are contrariwise not discussed in detail. ArchiMate as presented in [4] does not use attributes, whereas modelling tools that support ArchiMate for example BiZZdesign Architect[1] allow to use profiles in order to describe classes. However this tool does not provide to model relations between attributes and is therefore not capable to fulfill the requirements as they have been stated in chapter 2.

3.2 MEMO Meta-language

Frank discusses multi-perspective enterprise modelling (MEMO) [3] as an approach to integrate different perspectives on the enterprise. These perspectives are represented in different languages e.g. the *ITML* [28] modelling IT systems and resources. The languages are based on a common meta-language, the *MEMO metamodelling language* (MML) [29]. This language offers the syntactic primitives and model elements that are needed to describe EA conceptual models of the different special purpose languages. Basically, MML is an object-oriented metamodelling language introducing the meta-concepts MetaEntity, MetaAssociationLink, and MetaAttribute to designate classes, associations, and properties. Thereby, "classic" capabilities of object-oriented metamodelling facilities are provided, e.g. cardinalities or primitive data-types. A particular extensions is the *intrinsic* meta-concept. Frank furthers the discussion of Atkinson and Kühne in [30], who elaborate on the difficulties of describing different meta-levels within an object-oriented metamodel. Any modelling facility pursuing the paradigm of object-orientation centers around the dichotomy of meta-level, i.e. the level of the meta-concepts, and instance-level, on which the objects reside. Modelling domains that supply more than one level of *ontological instantiation* can hence not directly be covered by object-oriented modelling facilities. In order to avoid that language users introduce conceptually unclear 'workarounds' to the metamodels as the *type-item*-pattern, Frank adopts the idea of the *deep-instantiation*. He distinguishes the concepts along their *potency* in terms of Atkinson and Kühne [30]. A concept of potency n is instantiated to a concept of potency $n-1$, whereas concepts of potency 0 designate objects, instantiated associations, and bound attributes. Normal meta-concepts are of potency 1. Intrinsic meta-concepts are of potency 2, which are instantiated to normal meta-concepts.

The mechanism of the intrinsic concept is used by Frank et al. for the *Score-ML*, a language for defining (business) indicator systems within enterprise models [31]. The Score-ML defines the concept of the SpecificIndicator that measures a ReferenceObject. Such reference object can be an arbitrary element in an metamodel underlying a particular enterprise modelling language, i.e. perspective on

[1] http://www.bizzdesign.com/products/architect.html

the enterprise. On the level of the indicator definition, the specific indicator can supply benchmarks for interpreting the particularValue of the indicator as measured at an instance of the reference object. Figure 1 shows the corresponding cutout from the metamodel of the Score-ML. Therein, we use the stereotype «i» to designate intrinsic meta-concepts.

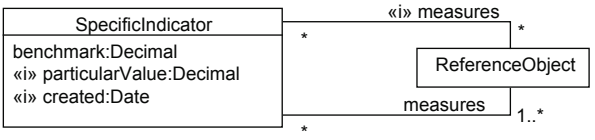


Fig. 1. Cutout of the indicator model of the Score-ML

In addition the Score-ML also supplies specialized relationship types that can be used to describe the dependencies between different indicators as well as between an indicator and a goal. One indicator can be computed from another indicator or can be similar to another one. These relationships are used during the design of the indicator system in different ways. Firstly, similar indicators can be exchanged with each other, in case one indicator is not measurable in the specific utilization context. Computational references are further operationalized during the design of an indicator system to actual computation rules that aggregate indicator values. Score-ML does not put particular emphasis on the notion of uncertainty, nor does it facilitate to introduce distributions acting as surrogates for actual indicator values.

3.3 Probabilistic Relational Models

As stated in the introduction of Section 3, PRMs serve as a guideline for the treatment of attributes, both in terms of support for uncertainty and relations between attributes. PRMs employ probabilistic reasoning to perform quantitative analysis of architecture properties. The main advantage of using PRMs instead of other probabilistic graphs, such as Bayesian networks [32] is their ability to also model objects of the world and their relations to each other in a manner similar to Entity-Relationship diagrams.

A PRM [21] specifies a template for a probability distribution over an architecture model. The template describes the metamodel for the architecture model, and the probabilistic dependencies between attributes of the architecture objects. A PRM, together with an instantiated architecture model of specific objects and relations, defines a probability distribution over the attributes of the objects. The probability distribution can be used to infer the values of unknown attributes, given the values of some attributes.

An architecture *metamodel* $\mathcal{M}$ describes a set of *classes*, $\mathcal{X} = X_1, \ldots, X_n$. Each class is equipped with a set of *descriptive attributes* and a set of *reference slots*. The set of descriptive attributes of a class X is denoted $\mathcal{A}(X)$. Attribute A of class X is denoted $X.A$ and its domain of *values* is denoted $V(X.A)$. For

example, a class `Business Process` might have the attribute *Availability*, with domain $\{Up, Down\}$. The set of reference slots of a class X is denoted $\mathcal{R}(X)$. We use $X.\rho$ to denote the reference slot ρ of class X. Each reference slot ρ is typed with the *domain type* $Dom[\rho] = X$ and the *range type* $Range[\rho] = Y$, where $X; Y \in \mathcal{X}$. A slot ρ denotes a relation from X to Y. Slots are similar to the relations in Entity-Relationship diagrams. For example we might have a class `Business Process` with the reference slot *uses* whose range is the class `Business Application`. For each reference slot ρ we have an inverse reference slot ρ^{-1} denoting the inverse relation. In the prior example, the class `Business Application` has an inverse reference slot $uses^{-1}$ to `Business Process`.

An architecture *instantiation* $\mathcal{I}$ (or an architecture model) specifies the set of objects of each class, the values for the attributes, and the references of the objects. It specifies a particular set of data objects, business processes, business applications, etc., along with their attribute values and references. We also define a *relational skeleton* σ_r as a partial instantiation which specifies the set of objects in all classes as well as all the reference slot values, but not the attribute values.

A probabilistic relational model Π specifies a probability distribution over all instantiations $\mathcal{I}$ of the metamodel $\mathcal{M}$. This probability distribution is specified similar to a Bayesian network [32], which consists of a qualitative dependency structure and associated quantitative parameters.

The *qualitative* dependency structure is defined by associating each attribute $X.A$ a set of parents $Pa(X.A)$ through attribute relations. Each parent of $X.A$ is defined as $X.\tau.B$ where $B \in \mathcal{A}(X.\tau)$ and τ is either empty, a single slot ρ or a sequence of slots $\rho_1, \ldots, \rho_k$ such that for all i, $Range[\rho_i] = Dom[\rho_{i+1}]$. In our example, the attribute $BusinessProcess.Availability$ may have $BusinessApplication.Availability$ (i,e. $Pa(BusinessApplication.Availability) = BusinessProcess.Uses^{-1}.Availability$) as parent, thus indicating that the Availability of an `Business Process` depends on the availability performed by the `Business Application`, which realize the `Business Process`. Note that $X.\tau.B$ may reference a set of attributes rather than a single one. In these cases, we let A depend probabilistically on some aggregated property over those attributes, such as the logical operations MIN or MAX. Considering the *quantitative* part of PRMs, given a set of parents for an attribute, we can define a local probability model by associating a *conditional probability distribution* (CPD) with the attribute, $P(X.A|Pa(X.A))$. We can now define a PRM Π for a metamodel $\mathcal{M}$ as follows. For each class $X \in \mathcal{X}$ and each descriptive attribute $A \in \mathcal{A}(X)$, we have a set of *parents* $Pa(X.A)$, and a CPD that represents $P_\Pi(X.A|Pa(X.A))$. Given a *relational skeleton* σ_r (i.e. a metamodel instantiated to all but the attribute values), a PRM Π specifies a probability distribution over a set of instantiations $\mathcal{I}$ consistent with σ_r:

$$P(\mathcal{I}|\sigma_r, \Pi) = \prod_{x \in \sigma_r(X)} \prod_{A \in \mathcal{A}(x)} P(x.A|Pa(x.A))$$

where $\sigma_r(X)$ are the objects of each class as specified by the relational skeleton σ_r. A PRM thus constitutes the formal machinery for calculating the probabilities of various architecture instantiations. This allows us to infer the probability

that a certain attribute assumes a specific value, given evidence of the rest of the architecture instantiation.

4 Extending the Meta Object Facility

The meta-object facility (MOF) [33] is an OMG standard that provides a multi-purpose metamodel for defining object-oriented models based on the UML infrastructure [34].[2] According to the findings in [16], MOF is prevalent as metamodel for defining EA information models with the exceptions as discussed in Section 3. With respect to requirements **R1**-**R3** MOF does not offer dedicated modelling mechanisms, neither for modelling existential uncertainty, uncertainty concerning values, nor dependencies between architectural properties. Below we introduce an extension to MOF that addresses the requirements. This extension builds on EMOF that defines basic concepts for object-oriented metamodels. Following Figure 2 displays the stack of meta-levels on which our modelling approach builds and indicates on which meta-level our extended EMOF resides.

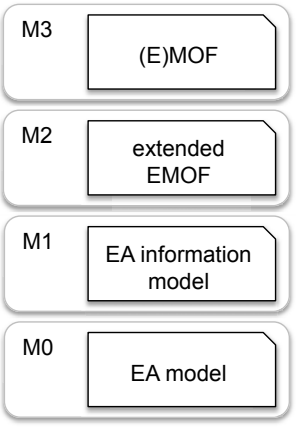


Fig. 2. Meta-modelling stack

Requirement **R1.1** is addressed by a model extension that allows to designate an existence probability for an instance of a particular class, i.e. for an InstanceSpecification in terms of MOF. An additional attribute exProbability can be used to describe that an instance has a lower existence probability than the default value 1. This value is retained to ensure compatibility with 'normal', i.e. non-extended, MOF models. The existential uncertainty with respect to relationships between instances as well as the uncertainty with respect to the value of a property are addressed in a uniform manner. This abides to the fact that in EMOF,

[2] MOF can also be used to 'boot-strap' itself, i.e. the concepts of MOF can be explained as instances of MOF concepts.

the concept property is used to describe both simple properties but also unidirectional relationships. The reference property.opposite is used to designate that two such unidirectional relationships constitute a bi-directional relationship in the sense of conceptual modelling. The value of a property at a particular instance is reflected in a slot which in turn is assigned a valueSpecification containing the actual value. Different types of value specifications are supplied by MOF, of which three are of particular interest for our extension. IntegerExpression and FloatExpression are used to specify values for Integer and Float properties, reflectively. We introduce these types of expressions as specializations to the generic ValueExpression of MOF that allows to specify untyped values. Both specialized expressions contain an attribute uncertainty that allows to specify, the degree of uncertainty in terms of the *variance*. We thereto assume two specific distributions for discrete and continuous properties, namely the binomial distribution and the normal distribution, correspondingly. The InstanceValue designates the instance specification assigned to a non-primitive, i.e. referencing, property. It further assigns a probability for the corresponding value assignment being correct. Using these mechanisms, we address requirements **R1.2** and **R2**, respectively. The following constraint is needed to ensure that properties representing bi-directional relationships are consistent with respect to their assigned probabilities:

```
context: InstanceValue
inv: InstanceValue->forall(i|self.owningSlot.definingFeature.opposite ==
i.owningSlot.definingFeature implies
self.proability == i.probability);
```

We fulfill **R3.1** through the introduction of the PropertyRelationship concept. A PropertyRelationship is a directed relationship between two properties that specifies influence from the source to the dependent property. The two properties may be owned by the same class or by transitively associated ones. In the latter case, the property relationship further specifies, which relationship path between the classes is influencing the dependency from the source to the target property. This means that the following constraints have to hold:

```
context: PropertyRelationship
inv: targetPath()->forAll(c|isSelfOrSub(c,sPath().at(tPath().indexOf(c))))
inv: sourceProperty.type.oclIsType(DataType)
inv: dependentProperty.type.oclIsType(DataType)
inv: dependencyPath->forAll(c|dependencyPath->count(c) == 1)
inv: sourceProperty <> dependentProperty
```

In these constraints we build on auxiliary operations:

```
context: AttributeRelationship
allSuperClasses(c:Class):Collection
allSuperClasses = c.superClass->collect(c|allSuperClasses(c)).flatten()
context: PropertyRelationship
isSelfOrSub(Class sub, Class super):Boolean
isSelfOrSub = (sub == super) or allSuperClasses(sub).contains(super)
context: PropertyRelationship
```

```
sPath():Sequence
sPath = dependencyPath->collect(p|p.type).prepend(sourceProperty.class)
context: PropertyRelationship
tPath():Sequence
tPath = dependencyPath->collect(p|p.class).append(dependentProperty.class)
```

Finally requirement **R3.2** is addressed through a link between PropertyRelationship and OpaqueExpression. The standard [13] defines an opaque expression as "an uninterpreted textual statement that denotes a (...) set of values when evaluated in a context". Using such expressions, we establish definitory dependencies between different properties, i.e. designate that and how one property can be computed from another.

Figure 3 shows the extended metamodel and highlights added or adapted meta-classes.

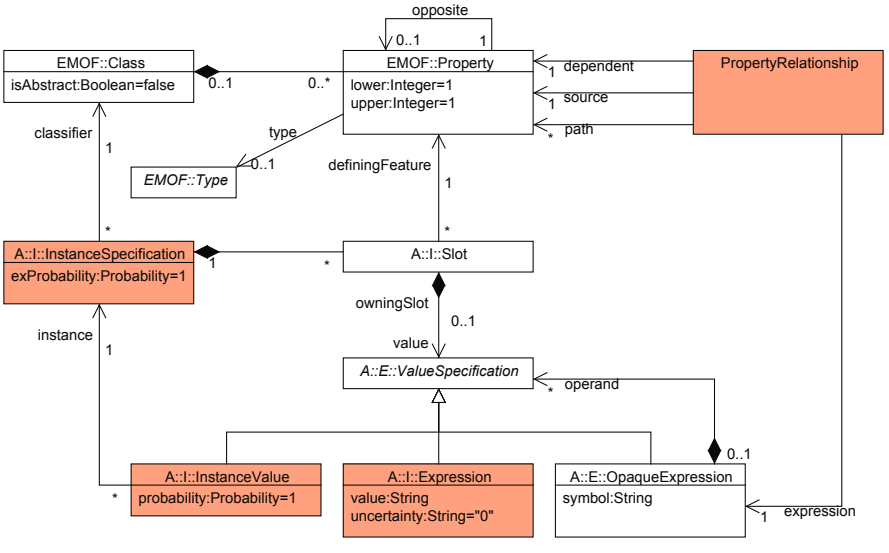


Fig. 3. Metamodel of the EMOF extension

5 Exemplifying the Language for EA Analysis

The basis for EA analysis are architecture models, which provide abstractions from the real world concepts. Dependencies between the modeled concepts play, as alluded to above, an important role in EA documentation and analysis. The subsequent example is concerned with metrics for application landscapes, which are an important part of the overall EA. The presented model, similar to the model introduced by Lankes in [35], discusses the availability of services offered by a business application in respect to the services used by the business application. Thereby, the aspect of failure propagation in the application landscape can be modeled on an abstract level.

The classes, attributes, and associations introduced therein are defined as follows:

BusinessApplication refers to a system, which is implemented in software, deployed at a specific location, and which provides support for at least one of the company's business processes. As consequence of the (not modeled) dependency of a business application to an underlying hardware device, the application has a certain level of availability (modeled as the probability for being available). In performing the business support, a business application may depend on other applications, which is modeled by two associations to the offered or used interfaces. The availability of the business application is thus dependent on the availability of the underlying hardware (not modeled) as well as on the availability of the interfaces used.

Interface is offered by a business application to provide a service for external use by one or more other applications or business processes. As a consequence of the provision by an application, the interface has an availability associated to the availability of the offering application.

BusinessProcess refers to a sequence of individual functions with connections between them. A business process as used in this model should not be identified with a single process step, but with high-level processes at a level similar to the one used in value chains. The execution of the process is dependent on the availability of the applications used.

Figure 4 shows the interplay between these concepts.

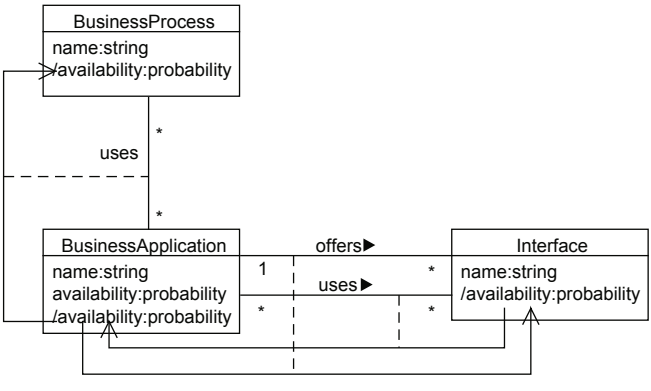


Fig. 4. Information model of the example

The dependencies between the values of the different availability attributes of the business application, the interface, and the business process class, which are informally defined in the textual descriptions, are visually indicated using the aforementioned notation for property relationships. In addition, the dependency between the availability of an interface and of the offering business application's availability can be expressed in simple terms, i.e. they are equal. A value specification can express this type of relationship. Figure 5 shows an instantiation of

the information model and calculates the probabilities for the availability given the corresponding uncertainties. Thereby, especially the uncertainty with respect to the interface LOG is taken into account. According to the current knowledge, this interface exists with a probability of 50% (exProbability = 0.5).

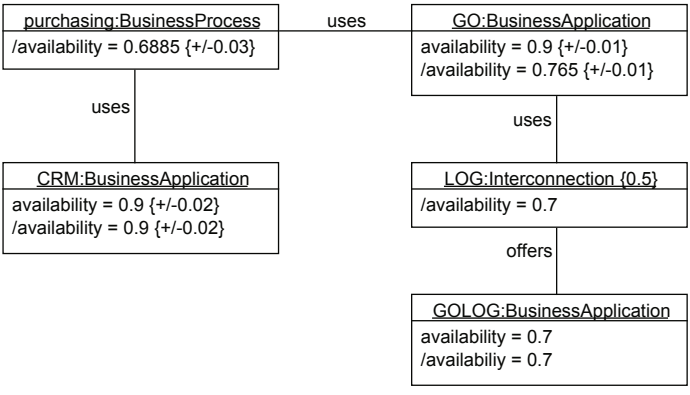


Fig. 5. Instance model of the example

6 Outlook

During the description of current states of the EA and the development of future states of the EA (cf. [36] for typical activities of EA management) two different types of uncertainty have to be considered: uncertainty with respect to the described current state, i.e. is the gathered information correct, or uncertainty with respect to a future state, i.e. is the plan to come true or did we make the right assumptions with respect to the future evolution. Are the right projects selected to come true. Combinations of the aforementioned uncertainties are also possible. In the approach only the describe uncertainty is discussed.

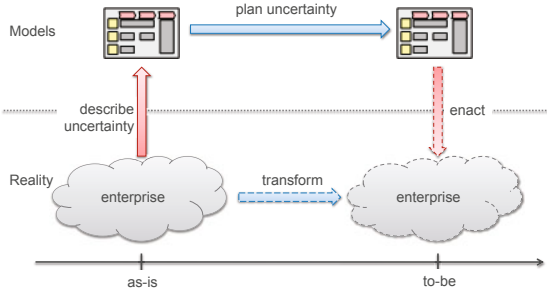


Fig. 6. Different types of uncertainty

Another interesting direction for future work targets the exact nature of the relationships between properties. The current model can describe that a relationship exists and can detail this relationship with an expression for their computation. On a more fine-grained level, we could distinguish between definitory dependencies and causal dependencies, of which the former are used to define a property, whereas the latter describe that a property depends on the value of another one. In both cases, it would also be possible to specify the direction of the dependency, i.e. to designate, whether dependent and depending property behave opposite or similar. While such specification is less expressive than the an actual computable expression, it can be used for qualitative analyses on the dependencies and the mutual reaction on changes.

Finally, the presented approach would benefit from an implementation. Therefore, existing tools (cf. [1]) can be extended to incorporate the MOF extension as presented in this paper. The implementation would further enable the evaluation of the meta-language in practice.

References

1. Buschle, M., Ullberg, J., Franke, U., Lagerström, R., Sommestad, T.: A tool for enterprise architecture analysis using the PRM formalism. In: CAiSE 2010 Forum PostProceedings (2010)
2. Buckl, S., Ernst, A.M., Lankes, J., Matthes, F.: Enterprise Architecture Management Pattern Catalog (Version 1.0, February 2008). Technical report, Chair for Informatics 19 (sebis), Technische Universität München, Munich, Germany (2008)
3. Frank, U.: Multi-perspective enterprise modeling (memo) – conceptual framework and modeling languages. In: 35th Hawaii International Conference on System Sciences (HICSS 2002), Washington, DC, USA, pp. 1258–1267 (2002)
4. Lankhorst, M.M.: Enterprise Architecture at Work: Modelling, Communication and Analysis, 2nd edn. Springer, Heidelberg (2009)
5. Dern, G.: Management von IT-Architekturen (Edition CIO). Vieweg, Wiesbaden (2006)
6. Niemann, K.D.: From Enterprise Architecture to IT Governance – Elements of Effective IT Management. Vieweg+Teubner, Wiesbaden, Germany (2006)
7. The Open Group: TOGAF "Enterprise Edition" Version 9 (2009), http://www.togaf.org (cited 2010-02-25)
8. Department of Defense (DoD) USA: DoD Architecture Framework Version 2.0: Volume 1: Introduction, Overview, and Concepts – Manager's Guide (2009), http://www.defenselink.mil/cio-nii/docs/DoDAF%20V2%20-%20Volume%201.pdf (cited 2010-02-25)
9. 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)
10. Murer, S., Worms, C., Furrer, F.J.: Managed evolution. Informatik Spektrum 31(6), 537–547 (2008)
11. Buckl, S., Ernst, A.M., Lankes, J., Schneider, K., Schweda, C.M.: A pattern based approach for constructing enterprise architecture management information models. In: Oberweis, A., Weinhardt, C., Gimpel, H., Koschmider, A., Pankratius, V. (eds.) Wirtschaftsinformatik 2007, Karlsruhe, Germany, pp. 145–162. Universitätsverlag, Karlsruhe (2007)

12. Kurpjuweit, S., Winter, R.: Viewpoint-based meta model engineering. In: Reichert, M., Strecker, S., Turowski, K. (eds.) 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 2007), Bonn, Germany. LNI, pp. 143–161. Gesellschaft für Informatik (2007)
13. Object Management Group (OMG): Omg unified modeling language™ (omg uml), superstructure – version 2.3 (formal/2010-05-05) (2010)
14. Object Management Group (OMG): Meta object facility (mof) core specification, version 2.0 (formal/06-01-01) (2006)
15. Buckl, S., Matthes, F., Schweda, C.M.: A meta-language for EA information modeling – state-of-the-art and requirements elicitation. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) BPMDS 2010 and EMMSAD 2010. Lecture Notes in Business Information Processing, vol. 50, pp. 169–181. Springer, Heidelberg (2010)
16. Buckl, S., Matthes, F., Schweda, C.M.: A meta-language for EA information modeling – state-of-the-art and requirements elicitation. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) BPMDS 2010 and EMMSAD 2010. Lecture Notes in Business Information Processing, vol. 50, pp. 169–181. Springer, Heidelberg (2010)
17. Johnson, P., Lagerström, R., Närman, P., Simonsson, M.: Enterprise architecture analysis with extended influence diagrams. Information Systems Frontiers 9(2), 163–180 (2007)
18. Aier, S., Buckl, S., Franke, U., Gleichauf, B., Johnson, P., Närman, P., Schweda, C., 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)
19. Becker, S., Koziolek, H., Reussner, R.: The Palladio component model for model-driven performance prediction. Journal of Systems and Software 82(1), 3–22 (2009)
20. Närman, P., Buschle, M., König, J., Johnson, P.: Hybrid Probabilistic Relational Models for System Quality Analysis. In: 14th IEEE International Enterprise Distributed Object Computing Conference (EDOC), pp. 57–66. IEEE, Los Alamitos (2010)
21. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: International Joint Conference on Artificial Intelligence, Stockholm, Sweden, vol. 16, pp. 1300–1309. Citeseer (1999)
22. Sommestad, T., Ekstedt, M., Johnson, P.: A probabilistic relational model for security risk analysis. Computers & Security 29(6), 659–679 (2010)
23. König, J., Nordstrom, L., Ekstedt, M.: Probabilistic Relational Models for assessment of reliability of active distribution management systems. In: 2010 IEEE 11th International Conference on Probabilistic Methods Applied to Power Systems (PMAPS), pp. 454–459. IEEE, Los Alamitos (2010)
24. Buckl, S., Franke, U., Holschke, O., Matthes, F., Schweda, C., Sommestad, T., Ullberg, J.: A pattern-based approach to quantitative enterprise architecture analysis. In: 15th Americas Conference on Information Systems, AMCIS (2009)
25. Lankhorst, M.M.: Introduction to enterprise architecture. In: Lankhorst, M. (ed.) Enterprise Architecture at Work, Springer, Heidelberg (2005)
26. Jonkers, H., van den Berg, H., Iacob, M.E., Quartel, D.: Archimate extension for modeling togaf's implementation and migration phases (2010), http://www.bizzdesign.com/index.php/component/docman/doc_download/18-archimate-extension-for-modeling-togafs-implementation-and-migration-phases
27. Halpin, T.: Object-role modeling (ORM/NIAM). In: Handbook on Architectures of Information Systems, pp. 81–103 (2006)

28. Kirchner, L.: Eine Methode zur Unterstützung des IT-Managements im Rahmen der Unternehmensmodellierung. PhD thesis, Universität Duisburg-Essen, Berlin, Germany (2008)
29. 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)
30. Atkinson,Colin, Kühne,Tomas: Reducing accidental complexity in domain models. Software and Systems Modeling, 345–359 (2007)
31. 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, Saarbrücken, Germany, November 27-28 (2008)
32. Jensen, F.V., Nielsen, T.D.: Bayesian networks and decision graphs. Springer, New York (2007)
33. Object Management Group (OMG): Meta object facility (mof) specification, version 2.0 (2006)
34. Object Management Group (OMG): Omg unified modeling language^{TM}(omg uml), infrastructure – version 2.3 (formal/2010-05-03) (2010)
35. Lankes, J.: Metrics for Application Landscapes – Status Quo, Development, and a Case Study. PhD thesis, Technische Universität München, Fakultät für Informatik, Munich, Germany (2008)
36. Buckl, S., Matthes, F., Schweda, C.M.: Towards a method framework for enterprise architecture management – a literature analysis from a viable system perspective. In: 5th International Workshop on Business/IT Alignment and Interoperability, BUSITAL 2010 (2010)

Towards an Investigation of the Conceptual Landscape of Enterprise Architecture*

D.J.T. van der Linden[1,2], S.J.B.A. Hoppenbrouwers[2], A. Lartseva[3], and H.A. (Erik) Proper[1,2]

[1] Public Research Centre Henri Tudor, Luxembourg, Luxembourg
{dirk.vanderlinden,erik.proper}@tudor.lu
[2] Radboud University Nijmegen, The Netherlands
stijnh@cs.ru.nl
[3] Donders Institute for Brain, Cognition and Behaviour, Nijmegen, The Netherlands
a.lartseva@fcdonders.ru.nl

Abstract. In this paper we discuss our preliminary work on clarifying the conceptual landscape of Enterprise Architecture. We do so to aid in the integration of conceptual models originating from different communities (of language users, concerns, viewpoints etc.). We propose that discovering the basic ontological structure used by these communities is necessary for the effective integration of models, and that different communities have a distinguishable different central understanding of some categories in their ontology. Our initial results include the description and categorization analysis of several languages and methods used in EA (as used by their creators), which suggest a prototype structure reflecting a community's focus.

Keywords: enterprise architecture, ontology, model, integration.

1 Introduction

Enterprise Architecture (EA) involves (the modeling of) many aspects, e.g. (business) processes, value exchanges, specifications, implementations, deployments and so on. These aspects are often handled by experts, who use specific languages or methods, rife with their own specialized terminology. As a result, these models are often created and maintained by separate communities and cannot trivially be related to each other.

* This paper results from the FNR ASINE project of CRP Henri Tudor and the Agile Service Development (ASD) project (http://www.novay.nl/okb/projects/agile-service-development/7628), a collaborative research initiative focused on methods, techniques and tools for the agile development of business services. The ASD project consortium consists of BeInformed, BiZZdesign, CRP Henri Tudor, HU University of Applied Sciences Utrecht, IBM, Novay, O&i, PGGM, RuleManagement Group, Radboud University Nijmegen, TNO, Twente University, Utrecht University, and Voogd & Voogd. The project is part of the program Service Innovation & ICT of the Dutch Ministry of Economic Affairs, Agriculture and Innovation.

T. Halpin et al. (Eds.): BPMDS 2011 and EMMSAD 2011, LNBIP 81, pp. 526–535, 2011.

A holistic perspective on the enterprise requires the integration of all specific views and related representations, which is a difficult endeavor. That integration presents a major issue is demonstrated by the amount of research and proposed methods for it [3,12,17]. What can be distilled from this research as a general trend is that integration requires that we respect the existence of the individual communities, and that we do not attempt to fix the issue by changing them. Instead we need to look towards how we (architects) deal with their models and products.

Efforts to integrate modeling are not new. Information system and model integration is covered by a staple of computing science efforts, yet most of them have focused on syntactical, and to a lesser extent, semantic integration. What is still missing, or has had less attention paid to it is local differences in semantics, comparable to 'local dialects' wielded by aspect communities.

To follow this line, we need a clearer understanding of what specific communities mean with their models, i.e. we need to reason about the meaning instead of the words and phrases they use to represent them. For a given aspect this means reasoning about a community's shared concepts – their conceptual landscape. We can use Discourse Communities for this (cf. [9]), which can function as a delimiter of the shared conceptual landscape. Perelman and Olbrechts-Tyceta [20] reason how meaning can only be understood in context of those shared factors:

> "All language is the language of community, be this a community bound by biological ties, or by the practice of a common discipline or technique. The terms used, their meaning, their definition, can only be understood in the context of the habits, ways of thought, methods, external circumstances, and tradition known to the users of those terms."

We propose that the users of some given modeling language or method in some typical context of use constitute such a community. Their shared habits (i.e. way and style of modeling) and ways of thinking (i.e. focus and construct availability), whether they existed prior to, or arose by influence of the language or method, are the differentiating factors from other (modeling) discourse communities, and as such can be used to find their shared common meaning.

In order to study these groups we have to represent their conceptual landscapes. This can be done best with ontology, as it is exactly that: a "formal explicit specification of shared conceptualization" [8]. However, it is necessary to clarify the structure and nature of ontology as it is often misused in computing. This specification must be understood as a representation or discovery of that conceptualization, and not a constructed or engineered optimal solution. Indeed, ontology in its original meaning concerns the (socially-constructed) categories of existence used by some group of people, and thus functions as a primer of what exists for these people in the 'real' world.

Where much existing work goes wrong is in not fully acknowledging the significance of categorization claims and commitments inherent in ontology (cf. Wyssusek's work on clarifying the mismatched basis of the Bunge-Wand-Weber ontology for information systems [26]). Furthermore, it is not uncommon for ontologies to be simply based on terminological categorization (cf. Uschold *et al.*'s

"the Enterprise Ontology" [25]) with no explicit regard to conceptual categorization. Most worrying is that (parts of) standard ontologies (e.g. Cyc, SUMO, Bunge-Wand-Weber) are used for a specific aspect with little regard to adapting them to their conceptual needs. This causes them to be stipulative in nature (i.e. we say or assume that people will see the world as dictated by some ontology) instead of descriptive (i.e. we find that people see the world in terms of their ontology) in describing the specific aspect of the enterprise.

When we adopt a descriptive approach and set out to discover 'the' ontology of some aspect, we need to keep in mind the structure of the categories and concepts therein. While most methods forgo this and implicitly assume a classical, atomistic structure (i.e. concepts are this or that), we find it more fitting to apply a graded approach, as exemplified by Prototype Theory. It holds that categorization does not occur on a strict comparison of necessary or sufficient properties, but that new potential members are compared to the most central (i.e. best exemplar) member of that category (the prototype). As such there is no perfect equality in a category but a graded structure where, for a given community, some members are better examples of some category than others.

This theory was initially proposed by Rosch [21], adopted by many others and used in a variety of investigations. It has been demonstrated in the categorization of abstract objects [1], and research has shown that events are categorized distinctly as well [14]. Because of this we can apply Rosch's theory to our concerns and the related discourse communities. What a prototype view adds is the ability to capture discourse community specific differences in interpretation, i.e. to find out whether users of some specific language or method share a different general conceptualization of a given category than other communities.

A descriptive ontology can then be used to resolve integration issues on a local semantic level by explicitly mapping each used word or phrase to the actual category of being. As such, it serves as a connecting layer between each community's models and the shared conceptualization discourse across communities. If we know what a word, phrase or given meaningful element in a model originating from some community actually means, we can know whether those elements can be integrated or not.

However, to get to this stage several key steps are necessary. The discourse communities have to be found and defined in such a way that they and their models can be studied. The contents of their ontology (i.e. the way they see the world) have to be found and defined and several verification cases have to be passed to study the correctness (and exhaustiveness) of any integrated model.

Hence, the contribution of this paper is twofold: discovering the initial ontological structure of the conceptual landscape that is shared amongst the different communities, and finding a starting point for prototype investigation by offering a set of discriminatory properties which help find the central members of a community's ontological categories. We believe this line of work will eventually lead to empirical data than can be used as input for other frameworks concerned with similar issues, such as the SEQUEL framework [11] and its 'social quality' aspect, i.e. the agreement (or Semantic Reassurance [9]) between modelers.

The remainder of this paper is structured as follows. In Section 2 we discuss our approach to studying the terms and meanings used by discourse communities and how we selected the languages and methods to study. In Section 3 we show the results of this categorization process and argue that they demonstrate a preference of different communities for different best examples of some categories: prototypes. Finally, we summarize our contribution in Section 4 and discuss the future work we will undertake to further validate our hypothesis.

2 Approach

In this section we discuss our approach to finding the discourse communities to study and the procedures used to analyze the data originating from this.

2.1 Selecting the Discourse Communities

We selected a variety of languages and methods originating both from academia and industry to ensure a wide pragmatic range in the respective language communities (albeit focused on the Western world). Where possible we based ourselves on official specifications (i.e. documents approved by standards management groups such as the OMG). If no such standardization or widely agreed upon specification existed, we based ourselves on the most complete and widely used information. An overview of the languages and methods selected as input material is given in Table 1. As we aimed to keep the analysis manageable in the initial phases there were many, more comprehensive, languages (e.g. MEMO, SysML, USDL etc.) and methods (e.g. TOGAF, IAF, MDA, etc.) that we did not include. Instead, we aimed to keep the input focused on languages and methods dealing with specific aspects, and not on comprehensive methods that attempt to solve the integration issue by stipulatively defining a shared vocabulary and way of working.

2.2 Categorization

The lexical representations of an aspect's units of meaning (i.e. a language's constructs or a method's given terminology) were gathered and classified according to their source of origin. Care was taken to prevent unwanted loss of information (i.e. collapsing multiple homonyms into one before the categorization process). During an iterative process we denoted (and subsequently refined) the (aspect-specific) categories that emerged from the meanings given for each term. These initially led to many specialized (almost atomic) categories, such as ACTOR[1], ACTORS, ARTIFICIAL ACTOR and so on. These were collapsed into the most common denominator (e.g. ACTOR) so that it would be possible to

[1] In order to avoid confusion between category and the words and phrases that populate it, a category will always be referred to in SMALL CAPS, while its population is shown in *italic*.

study whether prototype effects occurred within these categories for different communities. Finally each lexical representation was categorized into the final set of categories based on the meaning defined in the discourse community's specifications (e.g. *end event* being a typical RESULT before being considered a kind of EVENT in BPMN). In doing so we found a categorization (see Table 2) wide enough to fit the explicit differences found in the source material, while still remaining minimal in the amount of categories. Furthermore, the categorization is limited to the lexical representations of those constructs describing entities, as we found it undesirable to (superficially) cloud the analysis for the dubious virtue of including the many existing relationships within EVENT category.

Finally, we attempted to find possible clashing interpretations that could arise during integration and assigned them to a specific discriminant. We did so by means of introspective Semantic Differential analysis [18], i.e. looking for descriptions that would have both matches for themselves and their antonyms. For instance, the word *actor* could, depending on the discourse community, both be used to represent a human or non-human concept (for example, a person or a computer). From that clash it is deducible that 'human-ness' is a discriminating factor. The results of this analysis and the set of discriminants we found so far is shown in Table 3.

3 Discussion

The results of our analysis hint both at a common, shared categorization of concepts between different aspects of EA and at a difference in preferred, or central terminology. Thus, there seems to be a common conceptual ground that exhibits prototypical structure correlated to specific communities.

Some examples of this are members of the ACTOR category. In e3Value an *actor* is necessarily a being with some very specific properties (e.g. economic independence), while in a language often used with it, i*, it can also be other things, like a computer. Thus, while both communities share the common conceptual ground of an actor being some active entity, someone in the e3Value community would quickly associate it with what they see as economically independent beings (i.e. humans), while someone in the i* community can be expected to have a more generic abstract association.

Another example concerns members of the RESOURCE category. A language with a focus on deployment of software has many words for describing *hardware* that is needed, and as such a speaker of ADeL, ITML or other languages with comparable focus would readily associate RESOURCE with material objects that are needed to do something. Someone with other concerns, say architecture as expressed in ArchiMate, or more abstract process design like BPMN, would more quickly associate RESOURCE with some *information*, *knowledge*, or *environmental data* (i.e. immaterial objects) needed for something to be produced.

Yet more examples can be found in the RESTRICTION category, as many members thereof differ in how they deal with 'necessity' of compliance to some restriction. On the one hand, languages tasked with implementation or technical

Table 1. The discourse communities we used as the initial input for our analysis

Source	Aspect	Reason for inclusion
ArchiMate [23]	Architecture	A standard for EA modeling, it is one of the more expansive sources of EA-specific terminology.
ADeL [19]	Software deployment	The Application Deployment Language aims at describing (and validating) deployment of IT, which is an aspect that is missed in many other languages.
BPMN [16]	Processes	The Business Process Modeling Notation is a standard for process modeling and useful as a source as its language community is heavily involved with, and promotes abstract process thinking.
e3Value [5]	Value exchanges	While similar to other process thinking languages, e3Value has a very specific economic viewpoint, offering terminology specialized for value exchanges.
GRL [13] & i* [6] & KAOS [2]	Specifications	Goal-oriented Requirements Language, i* and Knowledge Acquisition in automated specification are widely used languages/methods used in goal thinking.
ITML [24]	Implementation and deployment	The IT Modeling Language offers terminology specific to those implementing and deploying software, which, as said earlier, many other languages lack.
ARIS [22]	Architecture	The Architecture of Integrated Information Systems is a framework whose community's terminology is on the intersection of process modeling and EA.
Balanced Scorecard [10]	Performance	An analysis framework intended for more than just (technical) modelers, it is a source of commonly used terminology by an important language community within EA; the not-necessarily technically inclined.
Game theory [15]	Behavior	Von Neumann's original concept and its terminology is often used in lieu of more 'standard' modeling languages when dealing with economic aspects.
RBAC [4]	Security	The terminology defined by Role-based access control is a near defacto-standard way of describing security issues, even outside of technical descriptions thereof.
VPEC-T [7]	Architecture (analysis)	The "Lost in Translation" approach offers an analysis framework that can be used by more than just those with technical expertise. For this reason it is worthwhile to take the terminology and its respective language community of non-technical users into account.

factors often deal with logical, computational laws, and therefore use words like *rule* in the alethic sense: restrictions that logically cannot be broken. On the other hand, those speakers dealing with human aspects often find themselves prescribing deontic rules or restrictions which can be violated, even though this is improper. It is obvious how, depending on the view of a speaker, his interpretation of the RESTRICTION category would be biased towards rules that are either alethic or deontic in nature.

Table 2. The categories and their members resulting from our analysis of the languages and methods. Numbers denote the originating community per Table 1's order.

Category	Class Members
ACTOR	unit[1], requirement unit[2], actor[1,4,5], role[1,5,10], collaboration[1], player[9], infrastructure/application component[1], device[1], application software[7], organizational unit[7], position[7,5], perspective[8], market segment[4], hardware role[6], software role[6], hardware[6], software[6], organizational role[6], environment/software agent[5]
EVENT	event[1,5,7,11], behavior[1], function[1,7], interaction[1], activity[3], task[3], business rule/service task[3], transaction[3], start event[3], intermediate event[3], value activity[4], value interface[4], value offering[4], connection[4], move[9], contribution[5], correlation[5], dependency[5], means-ends[5], control[5], goal refinement[5], monitor[5], operation[10], operationalisation[5], performance[5], operation[5], initiatives[8]
GOAL	goal[5,6,7], hard-goal[5], soft-goal[5], business goal[6], achieve goal[5], assignment[5], avoid goal[5], cease goal[5], expectation[5], maintain goal[5], requirement[5], consumer needs[4], belief[5], value[11], target[8]
PROCESS	organizational/infrastructure service[1], information/other service[7], service[1], IT service[6], process[3], sub/business/process flow[3], business process[6], dependency path[4], game[9], task[5]
RESOURCE	artifact[1,2], location[2,6], hardware[2,6] , cpu[2], hd[2], memory[2], software[2], value[1], data/business object[1], object[5], node[1], network[1,6], network device[6], representation[1], meaning[1], device[1], computer hardware[7], machine resource[7], environmental data[7], data input[3], input[5], value object[4], information[9], resource[5], content[11], value port[4]
RESTRICTION	interaction[1], contract[1], interface[1], message[7], catching[3], throwing[3], boundary[3,5], rule[9], decomposition[5], belief[5], priority[6], license[6], boundary condition[5], conflict[5], domain property[5], permission[10], value[11], policy[11], trust[11], (non)interrupting[3], strategy[9], measure[8], strategic objective[8]
RESULT	product[1], human/material/service output[7], data output[3], outcome[8], end event[3], payoff[9]
STATE	dependency element[4], dependency boundary[4]

Some counterintuitive categorizations can also be found in a number of communities. There is, for instance, a tendency for descriptions or types of things to be seen as specific kinds of those things. Some members of the ACTOR category like ARIS' *position*, ArchiMate's *organizational role* or RBAC's *role* are used by discourse communities not in the sense of *a role*, but as *some [actor] entity with the role of*. Similarly, depending on the focus of a community, words that are commonly seen as properties (for instance, *location*), can also be regarded as concrete concepts, like *location* being a (limited yet necessary) RESOURCE for those communities dealing with the (physical) deployment of machines.

While it is true that some of these differentiations of meaning between language communities might not be intended, factors like historical compromises in the design or implementation of a language (cf. ArchiMate's awkward handling of information as a physical entity [23], ch. 6.4), still allow for such differences to occur. In time those differences can become (unintentionally) entrenched in the

Table 3. Discriminants we found which can be used to differentiate category members

Discriminant	Explanation
Natural	Whether something was intentionally created (be it with a purpose or not) or naturally occurred. For example, an artifact versus a location or an actor versus some hardware.
Human	The human condition appears both in the sense of actors, for example a (human) actor or some software. It also seems to appear in composed structures, for example an organizational unit or market segment being necessarily subject to the human condition, a collaboration not so.
Composed	If something is composed of multiple parts (be it an aggregation or complex structure). For example, a single actor versus a organizational unit, or a cpu being a single resource, while a piece of hardware is seen as more than one.
Necessary	The difference between logical (im)possibility and probability. For example, a belief being a restriction that can be broken, while a (logical, or natural) rule is logically impossible to be broken.
Material	Something exists either as a physical, material object or is a (metaphysical) conception. For example, a materially existing resource can be some hardware, while a piece of information can be just as much of a resource without existing as a material object.
Intentional	Something is done, or provoked (be it with or without a reason) instead of spontaneously occurring. For example, an event can be a spontaneous occurrence while a transaction is always intentionally provoked
Vague	Something is either determined explicitly or is vague and left open for interpretation. For example, a hard goal or a soft goal.

conceptual landscape of a language and its speakers, especially if they are not quickly corrected. As a result, they will drift away from a shared understanding with once similar communities, and risk becoming a distinct community.

Knowing all this, it is obvious how important it is to discover these differences in meaning to know exactly what is meant by a model. In order for such information to be usable, let alone have a lasting usefulness, we need to explicitly map the results onto an ontology. As a starting point we have identified the basic categories (i.e. lower ontology) of the communities (of language creators) we analyzed. To figure out at a superficial level whether two communities mean the same thing by a word is thus to see if it at least belongs to the same category. This is a worthwhile endeavor as it can efficiently clear up confusion when the same word is used for different categories (i.e. homonymy). For instance, *hardware* is seen as a RESOURCE by some language communities and as an ACTOR by others.

More detailed mapping requires some more abstract, generic category of being, i.e. an upper ontology. The discovery of the upper ontology for this purpose is not a trivial matter, nor should an existing upper ontology be arbitrarily adopted. Discriminants (Table 3) can be used for this if we can find different combinations of discriminants that require different categories to exist and help to rule out existing upper ontologies that would not fit well.

For example, an e3Value *actor* and an ArchiMate *organizational unit* both belong to the ACTOR category. However, *actor* in this sense is naturally occurring, human and not composed, while *organizational unit* is not naturally occurring, human, and composed. Thus, while both communities speak of superficially the same thing, the mapping shows they are not interchangeable. Different words mapping to the same concept can be found in the same way. Take for example the RESULT category. An *outcome* in ARIS is often seen as a non-naturally occurring, non-materially intended 'thing', which would be the same as an *end event* is seen in BPMN. Thus, in this context the mapping shows the words are interchangeable.

To summarize, we have observed that different discourse communities seem to exhibit preference for the specific use of terminology, and thereby have different 'central' point(s) of a category. Thus, this work is a first critical step towards discovering the prototype structure of EA's conceptual landscape.

4 Conclusion and Future Work

In this paper we have argued for the use of descriptive, not stipulative, ontology as a means of promoting integration and a better shared understanding between (models originating from) different discourse communities in EA. We have created a starting point for such an EA-specific ontology by analyzing the language constructs and their meanings as they were defined by their creators. Furthermore, we have demonstrated several discriminating factors that set apart terminology within a given category, suggesting that these categories are based on prototypes reflecting their originating community's focus.

Future work will involve further investigation of the exact category structure for different discourse communities (e.g. process modelers, enterprise architects, software designers, etc.). We will do so by using the categories and discriminants from Tables 2 & 3 as a starting point for a Semantic Differential [18] experiment adapted to these groups. These experiments will result in datasets that can be used to characterize how different types of communities (and thus modelers) implicitly use the language constructs (different from their specified semantics), which can aid in integration by pointing out where semantic gaps are likely to occur. Further validation, depending on the outcomes of the first phase, may incorporate in-depth interviews and physiological conditioning experiments to also discriminate between conscious and subconscious semantic connections.

References

1. Adelson, B.: Comparing natural and abstract categories: A case study from computer science. Cognitive Science 9(4), 417 (1985)
2. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Sci. Comput. Program. 20, 3–50 (1993)
3. Dijkman, R.M., Quartel, D.A.C., van Sinderen, M.J.: Consistency in multiviewpoint design of enterprise information systems. Inf. Softw. Technol. 50(7-8), 737–752 (2008)

4. Ferrariolo, D., Cugini, J., Kuhn, R.: Role-based access control (rbac): Features and motivations. In: Proc. of the 11th Annual Computer Security Applications Conference (1995)
5. Gordijn, J., Yu, E., van der Raadt, B.: e-service design using i* and e3value modeling. IEEE Software 23, 26–33 (2006)
6. Grau, G., Horkoff, J., Yu, E., Abdulhadi, S.: i* guide 3.0. Internet (August 2007), http://istar.rwth-aachen.de/tiki-index.php?page_ref_id=67
7. Green, N., Bate, C.: Lost in Translation. Evolved Technologist (2007)
8. Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquis. 5, 199–220 (1993)
9. Hoppenbrouwers, S.J.B.A.: Freezing language: conceptualisation processes across ICT-supported organisations. Ph.D. thesis, Radboud University Nijmegen (2003)
10. Kaplan, R.S., Norton, D.P.: The balanced scorecard - measures that drive performance. Harvard Business Review, 71–79 (January-February 1992)
11. Krogstie, J., Jørgensen, H.D.: Quality of interactive models. In: Olivé, À., Yoshikawa, M., Yu, E.S.K. (eds.) ER 2003. LNCS, vol. 2784, pp. 351–363. Springer, Heidelberg (2003)
12. Lankhorst, M.M.: Enterprise architecture modelling–the issue of integration. Advanced Engineering Informatics 18(4), 205–216 (2004)
13. Liu, L., Yu, E.: From requirements to architectural design - using goals and scenarios. In: Software Requirements to Architectures Workshop, STRAW 2001 (2001)
14. Majid, A., van Staden, M., Boster, J., Bowerman, M.: Event categorization: A cross-linguistic perspective. In: Proc. of the 26th Annual Meeting of the Cognitive Science Society (2004)
15. von Neumann, J.: Theory Of Games And Economic Behavior. Princeton University Press, Princeton (1944)
16. Object Management Group: Business process model and notation (bpmn) ftf beta 1 for version 2.0. Internet (2010), http://www.omg.org/spec/UML/2.0/
17. Opdahl, A.L., Berio, G.: Interoperable language and model management using the ueml approach. In: Proc. of the 2006 International Workshop on Global Integrated Model Management, pp. 35–42. ACM, New York (2006)
18. Osgood, C.E., Suci, G.J., Tannenbaum, P.: The Measurement of Meaning. University of Illinois Press, Urbana (1957)
19. Patig, S.: Modeling deployment of enterprise applications. In: Soffer, P., Proper, E. (eds.) CAiSE Forum 2010. Lecture Notes in Business Information Processing, vol. 72, pp. 253–266. Springer, Heidelberg (2011)
20. Perelman, C., Olbrechts-Tyteca, L.: The New Rhetoric: A Treatise on Argumentation. University of Notre Dame Press (June 1969)
21. Rosch, E.: Natural categories. Cognitive Psychology 4(3), 328–350 (1973)
22. Scheer, A.-W., Nüttgens, M.: ARIS architecture and reference models for business process management. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, p. 376. Springer, Heidelberg (2000)
23. The Open Group: ArchiMate 1.0 Specification. Van Haren Publishing (2009)
24. Ulrich, F., Heise, D., Kattenstroth, H., Ferguson, D.F., Hadar, E., Waschke, M.G.: Itml: A domain-specific modeling language for supporting business driven it management. In: Proc. of the 9th OOPSLA Workshop on DSM (2009)
25. Uschold, M., King, M., Moralee, S., Zorgios, Y.: The enterprise ontology. The Knowledge Engineering Review 13(01), 31–89 (1998)
26. Wyssusek, B.: On ontological foundations of conceptual modelling. Scandinavian Journal of Information Systems 18, 63–80 (2006)

Author Index