Selmin Nurcan (Ed.)

LNBIP 107

IS Olympics: Information Systems in a Diverse World

CAiSE Forum 2011
London, UK, June 2011
Selected Extended Papers

Springer

Lecture Notes in Business Information Processing 107

Selmin Nurcan (Ed.)

IS Olympics: Information Systems in a Diverse World

CAiSE Forum 2011
London, UK, June 20-24, 2011
Selected Extended Papers

Springer

Volume Editor

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

ISSN 1865-1348 e-ISSN 1865-1356
ISBN 978-3-642-29748-9 e-ISBN 978-3-642-29749-6
DOI 10.1007/978-3-642-29749-6
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012935779

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

The CAiSE 2011 conference theme was linked with the coming Olympic and Paralympic Games, which bring together athletes from all continents to celebrate sporting excellence but also human diversity. Diversity is an important concept for modern information systems. Information systems (IS) are diverse by nature, as are the processes for constructing such systems, their developers, their users. It is therefore the responsibility of the IS engineering community to engineer information systems that operate in such a diverse world. During the two last decades, essential challenges have made their appearance in the area of IS related to engineering, quality and interconnectivity of IS.

The CAiSE 2011 Forum was a place within the CAiSE conference for presenting and discussing new ideas and tools related to IS engineering. Intended to serve as an interactive platform, the forum aimed at the presentation of fresh ideas, emerging new topics, controversial positions, as well as demonstration of innovative systems, tools and applications. The forum session at the CAiSE conference facilitated the interaction, discussion, and exchange of ideas among presenters and participants.

Two types of submissions were invited to the forum:

(1) Visionary short papers that present innovative research projects, which are still at a relatively early stage and do not necessarily include a full-scale validation.
(2) Demo papers describing innovative tools and prototypes that implement the results of research efforts. The tools and prototypes were presented as demos in the forum.

The 15 papers presented in this volume were carefully reviewed and selected from 46 submissions.

The CAiSE 2011 Forum received a record number of 46 submissions from 24 countries (Argentina, Australia, Austria, Brazil, Bulgaria, Canada, France, Germany, Hungary, Ireland, Israel, Italy, Japan, Latvia, Luxembourg, The Netherlands, Norway, Portugal, South Africa, Spain, Sweden, Switzerland, UK, USA). Among the submissions, 25 are demo papers and 21 are visionary papers.

The management of paper submission and reviews was supported by the EasyChair conference system. Selecting the papers to be accepted was a worthwhile effort. All papers received three reviews from the members of the Program Committee and the Program Board. Eventually, 23 high-quality papers were selected; among them 16 demo papers and 7 visionary papers. All of them were presented during the forum session in London, which attracted a big audience.

After the CAiSE 2011 Forum, authors of forum papers were invited to submit an extended version of their papers for the proceedings published as a Springer LNBIP volume. In all, 17 full papers were submitted. All papers again received three reviews from the members of the CAiSE 2011 Forum LNBIP Proceedings Editorial Committee. At the end of a two-round review process supported by EasyChair, this LNBIP volume presents a collection of 15 extended papers; among them 6 visionary papers that present innovative research projects, which are still at a relatively early stage and do not necessarily include a full-scale validation; and 9 demo papers describing innovative tools and prototypes that implement the results of research efforts.

As the CAiSE 2011 Forum Chair, I would like to express again my gratitude to the forum Program Board and the Program Committee for their efforts in providing very thorough evaluations of the submitted papers.

As the volume editor of the CAiSE 2011 Forum LNBIP proceedings, I would like to express my gratitude to the Proceedings Editorial Committee members for their sustainable efforts in providing very thorough evaluations of the submitted extended forum papers. I wish also to thank all authors who submitted papers to the forum LNBIP proceedings for having shared their work with us.

Last but not least, I would like to thank the CAiSE 2011 Program Committee Chairs and the Local Organizing Committee for their support.

February 2012 Selmin Nurcan

Organization

CAiSE 2011 Forum Chair

Selmin Nurcan	Université Paris 1 Panthéon-Sorbonne, France

Editorial Committee Members

João Paulo A. Almeida	Federal University of Espírito Santo, Brazil
Judith Barrios	Universidad de Los Andes, Venezuela
Fazli Can	Bilkent University, Turkey
François Charoy	Nancy Université, France
Maya Daneva	University of Twente, The Netherlands
Xavier Franch	Universitat Politecnica de Catalunya, Spain
Dragan Gasevic	Athabasca University, Canada
Stewart Green	University of the West of England, UK
Chihab Hanachi	Université Toulouse 1 Sciences Sociales, France
Evangelia Kavakli	University of the Aegean, Greece
Agnes Koschmider	Karlsruhe Institute of Technology, Germany
Sai Peck Lee	University of Malaya, Kuala Lumpur, Malaysia
Hui Ma	Victoria University of Wellington, New Zealand
Selmin Nurcan	Université Paris 1 Panthéon-Sorbonne, France
Naveen Prakash	MRCE, India
Manfred Reichert	The University of Ulm, Germany
Hajo Reijers	Eindhoven University of Technology, The Netherlands
Michael Rosemann	Queensland University of Technology, Australia
Samira Si-Said Cherfi	CNAM, France
Pnina Soffer	University of Haifa, Israel
Janis Stirna	University of Stockholm, Sweden
Arnon Sturm	Ben-Gurion University of the Negev, Israel
Jelena Zdravkovic	Royal University of Technology, Stockholm, Sweden

Program Board Members

Nacer Boudjilida	Nancy Université, France
Xavier Franch	Universitat Politecnica de Catalunya, Spain
Pnina Soffer	University of Haifa, Israel
Manfred Reichert	The University of Ulm, Germany
Michael Rosemann	Queensland University of Technology, Australia
Carson Woo	University of Toronto, Canada

Program Committee Members

João Paulo A. Almeida	Federal University of Espírito Santo, Brazil
Judith Barrios	Universidad de Los Andes, Venezuela
Fazli Can	Bilkent University, Turkey
François Charoy	Nancy Université, France
Maya Daneva	University of Twente, The Netherlands
Chiara Francalanci	Politecnico di Milano, Italy
Dragan Gasevic	Athabasca University, Canada
Stewart Green	University of the West of England, UK
Chihab Hanachi	Université Toulouse 1 Sciences Sociales, France
Evangelia Kavakli	University of the Aegean, Greece
Agnes Koschmider	Karlsruhe Institute of Technology, Germany
Hui Ma	Victoria University of Wellington, New Zealand
Sai Peck Lee	University of Malaya, Kuala Lumpur, Malaysia
Naveen Prakash	MRCE, India
Jan Recker	Queensland University of Technology, Brisbane, Australia
Hajo Reijers	Eindhoven University of Technology, The Netherlands
Samira Si-Said Cherfi	CNAM, France
Janis Stirna	University of Stockholm, Sweden
Arnon Sturm	Ben-Gurion University of the Negev, Israel
Jelena Zdravkovic	Royal University of Technology, Stockholm, Sweden

Table of Contents

Innovative Tools and Prototypes

A Tool for Automatic Enterprise Architecture Modeling 1
Markus Buschle, Hannes Holm, Teodor Sommestad, Mathias Ekstedt, and Khurram Shahzad

Creating Declarative Process Models Using Test Driven Modeling Suite 16
Stefan Zugal, Jakob Pinggera, and Barbara Weber

Discovering Hierarchical Process Models Using ProM 33
R.P. Jagadeesh Chandra Bose, Eric H.M.W. Verbeek, and Wil M.P. van der Aalst

Diagen: A Model-Driven Framework for Integrating Bioinformatic Tools 49
Maria José Villanueva, Francisco Valverde, Ana M. Levín, and Oscar Pastor Lopez

A Software Framework for the Automated Production of Schematic Maps 64
Joao Mourinho, Teresa Galvao, João Falcão e Cunha, Fernando Vieira, and Jose Pacheco

A Flexible System for Ontology Matching 79
DuyHoa Ngo, Zohra Bellahsene, and Remi Coletta

A CASE Tool to Support Automated Modelling and Analysis of Security Requirements, Based on Secure Tropos 95
Michalis Pavlidis, Shareeful Islam, and Haralambos Mouratidis

A Tool for Managing Evolving Security Requirements 110
Gábor Bergmann, Fabio Massacci, Federica Paci, Thein Than Tun, Dániel Varró, and Yijun Yu

Tool Support for Enforcing Security Policies on Databases 126
Jenny Abramov, Omer Anson, Arnon Sturm, and Peretz Shoval

Visionary Papers

Towards Agile Model-Driven Web Engineering 142
José Matías Rivero, Julián Grigera, Gustavo Rossi, Esteban Robles Luna, and Nora Koch

Evaluating Comprehension and Utilization of Variability Aspects in UML-Based Models 156
Iris Reinhartz-Berger, Arnon Sturm, and Arava Tsoury

OLAP on Complex Data: Visualization Operator Based on Correspondence Analysis 172
Sabine Loudcher and Omar Boussaid

Knowledge Dimension in Business Process Modeling 186
Ligita Businska and Marite Kirikova

When Process Mining Meets Bioinformatics 202
R.P. Jagadeesh Chandra Bose and Wil M.P. van der Aalst

Stepwise Context Boundary Exploration Using Guide Words 218
Naoyasu Ubayashi and Yasutaka Kamei

Author Index 235

A Tool for Automatic Enterprise Architecture Modeling

Markus Buschle, Hannes Holm, Teodor Sommestad,
Mathias Ekstedt, and Khurram Shahzad

Industrial Information and Control Systems, KTH Royal Institute of Technology,
Osquldas v. 12, SE-10044 Stockholm, Sweden
{markusb,hannesh,teodors,mathiase,khurrams}@ics.kth.se

Abstract. Enterprise Architecture is an approach which aims to provide decision support based on organization-wide models. The creation of these models is however cumbersome as multiple aspects of an organization need to be considered. The Enterprise Architecture approach would be significantly less demanding if data used to create the models could be collected automatically.

This paper illustrates how a vulnerability scanner can be utilized for data collection in order to automatically create Enterprise Architecture models, especially covering infrastructure aspects. We show how this approach can be realized by extending an earlier presented Enterprise Architecture tool. An example is provided through a case study applying the tool on a real network.

Keywords: Enterprise Architecture, Automatic data collection, Automatic instantiation, Software tool, Security Analysis.

1 Introduction

Enterprise Architecture (EA) is a comprehensive approach for management and decision-making based on models of the organization and its information systems. An enterprise is typically described through dimensions such as Business, Application, Technology and Information. [12]. These pictographic descriptions are used for system-quality analysis to provide valuable support for IT and business decision-making [6].

As these models are intended to provide reliable decision support it is imperative that they capture all the aspects of an organization which are of relevance. Thus, they often grow very large and contain several thousands of entities and an even larger number of relationships in between them. The creation of such large models is both time and cost consuming, as lots of stakeholders are involved and many different pieces of information have to be gathered. During the creation process the EA models are also likely to become (partly) outdated [1]. Thus, in order to provide the best possible decision support it needs to be ensured that EA models both are holistic and reflect the organizations current state.

Automatic data collection and model creation would be useful as this would decrease the modeling effort and increase the quality of the collected data.

In this paper we present how the Enterprise Architecture Analysis Tool [6] has been extended in order to automatically instantiate elements in EA models based on results from network scans. In comparison to other tools our implementation focuses on the Application and Technology layer of the organization. This information is gathered through an application of a vulnerability scanner that evaluates the network structure of an enterprise. Thereby attached network hosts and the functionality they provide can be discovered. Another difference is that the presented tool uses EA models for system-quality analysis, whereas commercial applications focus on modeling. As a running example we illustrate how a meta-model designed for cyber security analysis [16] can be (partly) automatically instantiated. The presented implementation is generic and can be used to support any kind of EA analysis.

The remainder of this paper is structured as follows. Related work is discussed in section two. Section three describes the components used to realize the implementation and introduces into the meta-model that is used as running example. Section four describes how the information, which was automatically collected, is used to instantiate the meta-model for security evaluation. Section five exemplifies the tool application on real data collected by scanning a computer network used for security exercises. In section six the presented tool and the underlying approach are discusses as well as future work is described. Finally section seven concludes the paper.

2 Related Works

In the EA community there are few initiatives focusing on the data collection process for model instantiation and maintenance. Among the most well-known frameworks data collection is almost completely left to the modeler to handle. There are a couple of tool vendors providing some support. However, it is required in most cases that the needed information is available somewhere else, implying that the data must be collected by someone at some point in time. In the academic EA community most researchers have put their focus on deriving principles and designing methods for model creation and maintenance. Few have implemented their ideas in working software tools. None claims to have the focus on automatic data collection.

2.1 EA Frameworks

There are many frameworks covering EA modeling e.g. [21,12,19]. However, few (none) of these describe and discuss the data collection process used when creating the architecture models. No practical help is presented in these frameworks regarding the data collection for As-is models or for checking the consistencies in continuously updated architecture models (maintaining the architecture).

2.2 EA Tools

In current EA tools some approaches addressing automatic data collection can be found. The most common way is to import models that are made in third party software. For example, BizzDesign Architect [3] can import data from office applications and with this data instantiate models. Thereby the automation aspect actually means that data is reused and does not need to be manually entered if it is already available. The interpretation of data documented in the third party software can however be resource- and time consuming, thus contradicting parts of the purpose with automatic data collection.

Other tools, such as for example Troux [20], allow the usage of SQL queries in order to load information from data bases. This approach focuses on the extraction of the data model and process descriptions, thereby the automatic creation of the information architecture as well as the business architecture.

Both approaches assume that the data entered, in the third party applications or data bases, is already available and updated. However, this data still needs to be manually collected in the first place before it can be used.

ARIS Business Architect for SAP [15] supports the reuse and import of SAP process models out of the SAP Solution Manager. Thereby modeling costs are reduced. The SAP process model does only cover certain parts of the complete EA. While other aspects of EA, such as infrastructure, are not considered.

[18] presents a software tool and methodology used for collecting architecture information. The tool collects data from project management systems and operational systems, and users can upload documents containing architecture information. This approach thus aims to provide automatic data collection for architecture models. The approach is however still time consuming. Since the tool proposed use .cvs-files the data documentation process still needs to be formalized.

2.3 EA Methods and Principles

[5] presents a Wiki-based approach to EA documentation and analysis. According to Buckl et al. companies who start an EA initiative usually do not have a pre-defined information model for this. Many companies start with regular spreadsheets or similar. Instead, Buckl et al. propose a Wiki-based approach for collecting and sorting the information needed for EA management. The main benefit with the Wiki-approach is that the data collection is distributed but still managed by a formal set up. Although the Wiki-based approach proposed seems interesting there is still a need for data collection to provide input to the Wiki.

In [4] an approach for handling change is proposed. The approach is called Living Models and it is based on theories of model based software development and EA management. The idea is to connect IT management, System Operation, and Software Engineering. According to the author three major research challenges have to be met in order to materialize this: 1) Provide a coherent view of the quality status of the systems. 2) Keep track of the quality status as the systems evolve over time. 3) Support the collaboration of stakeholders for achieving the necessary quality level. EA models can be used to achieve number one of the three challenges. Automatic data collection would be appropriate for challenge number two. In the paper Breu presents ten principles that are crucial for Living Models. Principle no 2 - Close Coupling of Models and Code states: "Models are generated out of the code (e.g. architecture models)". This would mean automatic generation of models at some architecture levels. Furthermore, Principle no 3 - Bidirectional Information Flow between Models and Code, focus on the idea that information from code can be used to build models as well as information in models can be used to generate code. Throughout the ten principles patterns and meta-model elements are discussed supporting these ideas. However, there is no tool today that can implement and use these principles yet.

[9] states that: "Architecture management should produce methodological results in the form of architecture artifacts such as models, standards, etc." Further they state that: "Obviously, it is not possible to assume a consistent EA at a specific point in time, which means there should be a prime focus on dealing with inconsistencies." The authors conclude that architecture models will drift and become inconsistent. They spend some effort on trying to address this problem as it has occurred. What they do not describe is how to collect the information needed to actually compare the real world with the as-is models or update the models with the correct data.

In [2] the focus is on the design of the EA. The main result is a framework for engineering driven EA design and a software tool implementation of this framework. There is no description of the data collection process for the instantiation of models. The design of an EA here basically means deriving the meta-model needed for an enterprise. The software tool implementation proposed is a tool incorporating the framework and meta-model. In [7] the focus is on model maintenance and the main finding reported in this publication is a discussion of the shortcomings of existing model maintenance approaches. The authors present a federated approach to deal with these shortcomings. However, there is no discussion regarding the data collection part of model maintenance.

2.4 Summary

There are few initiatives describing and discussing data collection for EA modeling. There are even less initiatives proposing an automatic data collection process. Most research publications related to the topic provide evidence that the topic is important and needs to be addressed.

3 Preliminaries

This section describes the three components that were combined in order to automatically create EA models that can be used for security analysis. In subsection 3.1 the vulnerability scanner NeXpose [14], which is used for data collection, is explained. Subsection 3.2 describes the Enterprise Architecture Analysis Tool that is used to generate the models and evaluate them with regards to security aspects. Subsection 3.3 briefly introduces CySeMoL, the used meta-model which is partly instantiated using the automated data collection. The overall architecture can be seen in figure 1.

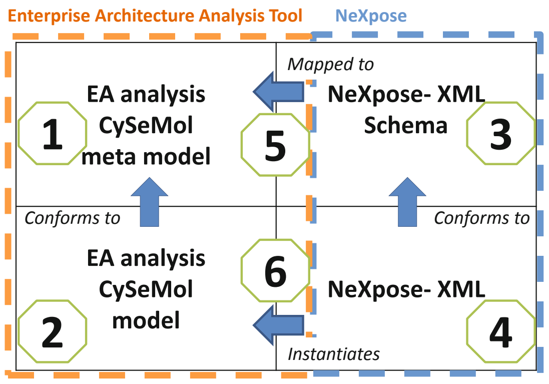

Enterprise Architecture Analysis Tool
NeXpose
Mapped to
EA analysis CySeMol meta model
1
5
NeXpose- XML Schema
3
Conforms to
Conforms to
EA analysis CySeMol model
6
NeXpose- XML
2
4
Instantiates

Fig. 1. The used architecture

3.1 NeXpose

The vulnerability scanner NeXpose was chosen in this project as it has demonstrated good results in previous tests [10]. A vulnerability scanner is a software tool which probes a network architecture for various vulnerabilities, for example, poor passwords and software flaws (e.g. unpatched software with known vulnerabilities).

NeXpose [14] is an active (i.e. it queries remote hosts for data) vulnerability scanner capable of both authenticated and unauthenticated scans. Authenticated scans involve providing the scanner with user accounts to hosts. They are typically less disturbing to normal operations and providing a higher degree of accuracy. However, it is not always the case that credentials are readily available for the individual(s) performing a scan.

NeXpose provides information regarding the network architecture in terms of all devices which are communicating over TCP or UDP, e.g. computers, firewalls and printers. The scanner identifies the operating systems or firmware that is running on the scanned devices and any services that are running. If the scanner is given credentials it is also able to assess all applications (and versions thereof) installed on a device and all user/administrator accounts on that device.

More security related functions of the scanner include that it can check for both software flaws and configuration errors. It is also capable of performing web application scans. NeXpose has approximately 53000 current signatures in its engine, with every signature corresponding to a certain vulnerability. NeXpose is also SCAP-compliant [11] and thus compliant with a suite of six commonly used protocols developed by the National Institute of Standards and Technology (NIST): i) Extensible Configuration Checklist Description Format (XCCDF),

ii) Open Vulnerability and Assessment Language (OVAL), iii) Common Platform Enumeration (CPE), iv) Common Configuration Enumeration (CCE), v) Common Vulnerabilities and Exposures (CVE) and vi) Common Vulnerability Scoring System (CVSS).

3.2 Enterprise Architecture Analysis Tool

In [6] we presented a tool for EA analysis. This tool consists of two parts to be used in succession. The first component allows the definition of meta-models to describe a certain system quality of interest (**1** in Figure 1). This is done according to the PRM formalism [8] in terms of classes, attributes, and relations between them. Thereafter an execution of the second component is done in order to describe an enterprise as an instantiated model (**2** in Figure 1), which is compliant to the previously defined meta-model. As the PRM formalism supports the expression of quantified theory the described enterprise can be analyzed with regards to the considered system quality described in the first component.

To use the results gained from NeXpose scans an extension of the tool was necessary. The result of NeXpose's scans can be exported to XML files (**4** in Figure 1), which are structured according to a schema definition file (XSD)[1] (**3** in Figure 1). We added the possibility to create mappings between XSD files and meta-models (**5** in Figure 1) in order to automatically instantiate the meta-model based on NeXpose's XML files (**6** in Figure 1). The used mapping is discussed in section 4.

3.3 CySeMoL

This paper exemplifies the mapping functionality by instantiating a subset of the meta-model of the Cyber Security Modeling Language (CySeMoL) [17]. This modeling language follows the abstract model presented in [16] and uses the PRM formalism to estimate the value of security attributes from an architecture model. Its meta-model covers both technical and organizational aspects of security and can be seen in Figure 2 (without any attribute relationships shown). CySeMoL includes 22 entities, each with various attributes and relations to other entities. It can require a lot of effort to model a scenario using the CySeMoL as a typical network contains multiple components e.g. computer systems, operating systems, services, password accounts, and application clients. Furthermore, each of these entities have security related aspects (i.e. attribute states) that needs to be modeled in order to achieve satisfactory prediction quality. The states of most of the attributes in the meta-model are defined by the states of their parents. However, the states of some variables need to be defined by the user of the model. For example whether there are vulnerabilities available for an operating system or a service. The required modeling effort could result in errors caused by the individual(s) performing the modeling.

[1] The XSD file (Report_XML_Export_Schema.xsd) is part of the NeXpose Community Edition that can be downloaded from http://www.rapid7.com

As such, it would be very valuable if parts of the CySeMoL meta-model could be automatically generated - it would save both modeling time and likely increase the accuracy of the resulting prediction model. Four entities, three entity relationships and the states of four attributes can be mapped to elements produced by NeXpose. These concepts, and the mapping as such, are discussed in section 4. While only a subset of the total number of entities and relations could be instantiated, this subset includes entities, relations and attributes which are of high multiplicity in enterprises, and thus require lots of effort to model.

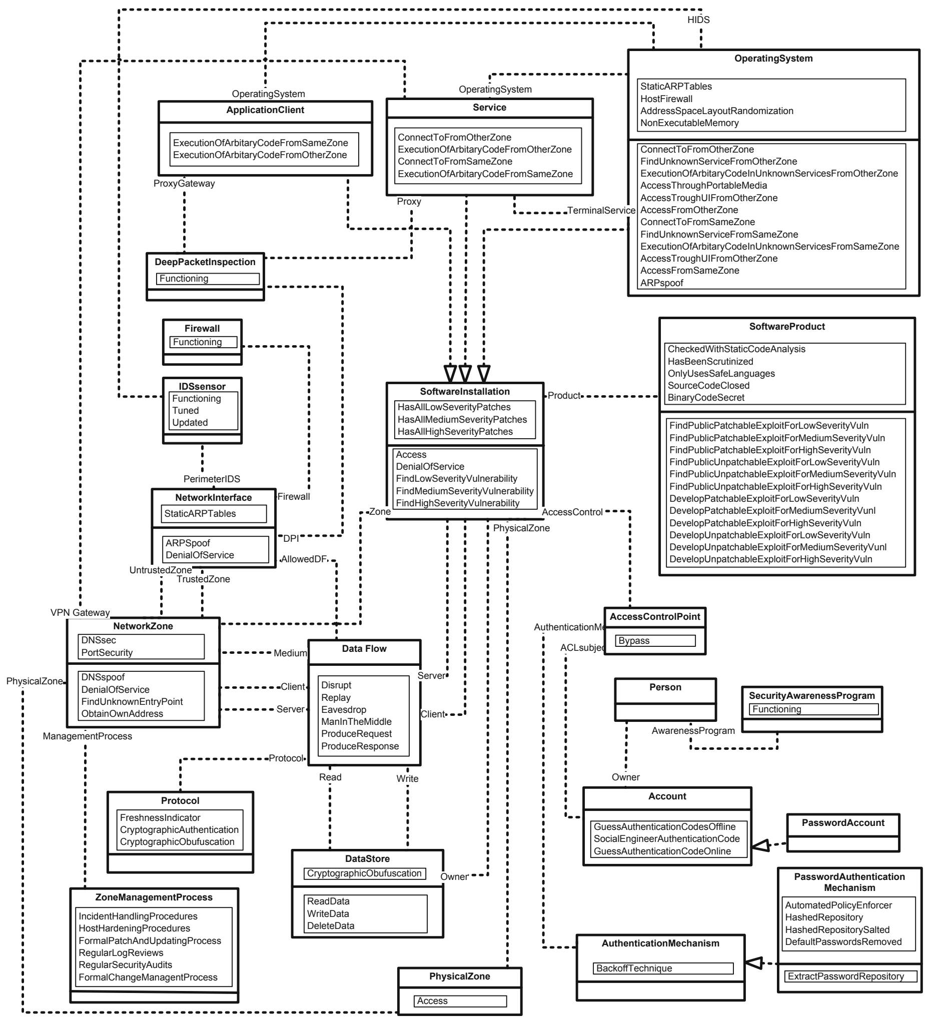

HIDS
OperatingSystem
StaticARPTables
HostFirewall
AddressSpaceLayoutRandomization
NonExecutableMemory
ConnectToFromOtherZone
FindUnknownServiceFromOtherZone
ExecutionOfArbitaryCodeInUnknownServicesFromOtherZone
AccessThroughPortableMedia
AccessTroughUIFromOtherZone
AccessFromOtherZone
ConnectToFromSameZone
FindUnknownServiceFromSameZone
ExecutionOfArbitaryCodeInUnknownServicesFromSameZone
AccessTroughUIFromOtherZone
AccessFromSameZone
ARPspoof
OperatingSystem
ApplicationClient
ExecutionOfArbitaryCodeFromSameZone
ExecutionOfArbitaryCodeFromOtherZone
OperatingSystem
Service
ConnectToFromOtherZone
ExecutionOfArbitaryCodeFromOtherZone
ConnectToFromSameZone
ExecutionOfArbitaryCodeFromSameZone
ProxyGateway
Proxy
TerminalService
DeepPacketInspection
Functioning
Firewall
Functioning
IDSsensor
Functioning
Tuned
Updated
SoftwareProduct
CheckedWithStaticCodeAnalysis
HasBeenScrutinized
OnlyUsesSafeLanguages
SourceCodeClosed
BinaryCodeSecret
FindPublicPatchableExploitForLowSeverityVuln
FindPublicPatchableExploitForMediumSeverityVuln
FindPublicPatchableExploitForHighSeverityVuln
FindPublicUnpatchableExploitForLowSeverityVuln
FindPublicUnpatchableExploitForMediumSeverityVuln
FindPublicUnpatchableExploitForHighSeverityVuln
DevelopPatchableExploitForLowSeverityVuln
DevelopPatchableExploitForMediumSeverityVunl
DevelopPatchableExploitForHighSeverityVuln
DevelopUnpatchableExploitForLowSeverityVuln
DevelopUnpatchableExploitForMediumSeverityVunl
DevelopUnpatchableExploitForHighSeverityVuln
SoftwareInstallation
HasAllLowSeverityPatches
HasAllMediumSeverityPatches
HasAllHighSeverityPatches
Access
DenialOfService
FindLowSeverityVulnerability
FindMediumSeverityVulnerability
FindHighSeverityVulnerability
Product
PerimeterIDS
NetworkInterface
StaticARPTables
ARPSpoof
DenialOfService
Firewall
DPI
AllowedDF
Zone
AccessControl
PhysicalZone
UntrustedZone
TrustedZone
VPN Gateway
NetworkZone
DNSsec
PortSecurity
DNSspoof
DenialOfService
FindUnknownEntryPoint
ObtainOwnAddress
PhysicalZone
Medium
Client
Server
Data Flow
Disrupt
Replay
Eavesdrop
ManInTheMiddle
ProduceRequest
ProduceResponse
Server
Client
Protocol
ManagementProcess
Read
Write
AuthenticationMe
ACLsubjec
AccessControlPoint
Bypass
Person
SecurityAwarenessProgram
Functioning
AwarenessProgram
Owner
Account
GuessAuthenticationCodesOffline
SocialEngineerAuthenticationCode
GuessAuthenticationCodeOnline
PasswordAccount
PasswordAuthentication Mechanism
AutomatedPolicyEnforcer
HashedRepository
HashedRepositorySalted
DefaultPasswordsRemoved
ExtractPasswordRepository
AuthenticationMechanism
BackoffTechnique
Protocol
FreshnessIndicator
CryptographicAuthentication
CryptographicObufuscation
DataStore
CryptographicObufuscation
ReadData
WriteData
DeleteData
Owner
ZoneManagementProcess
IncidentHandlingProcedures
HostHardeningProcedures
FormalPatchAndUpdatingProcess
RegularLogReviews
RegularSecurityAudits
FormalChangeManagentProcess
PhysicalZone
Access

Fig. 2. An overview of CySeMoL without attribute relations shown

4 The Mapping

This section describes the concepts of CySeMoL that were possible to map to the output of NeXpose, and how these concepts are defined by NeXpose. This mapping was done based on the XSD file that describes the structure of the vulnerability scanning reports produced by NeXpose. Furthermore, the mapping is based on the viewpoint of CySeMoL as this is the focus of the study – to see how much of its contents that can be automatically generated. All of the entities described in this chapter were possible to relate as shown in Figure 2.

4.1 Operating System

The CySeMoL entity OperatingSystem and three of its attributes are possible to automatically instantiate (cf. Figure 3. These concepts are described below. As can be seen, all information required by CySeMoL regarding these aspects are fulfilled.

OperatingSystem. *CySeMoL*: The software system which other software is deployed on. Examples include Windows XP SP2 (with a specific set of patches), VMware vSphere 5.1.0 and printer firmwares (and versions thereof). *NeXpose*: The same definition as CySeMoL, abstractly described as the Operating System of a Node. Gathered information includes e.g. OS type and version, and the IP adress of the system.

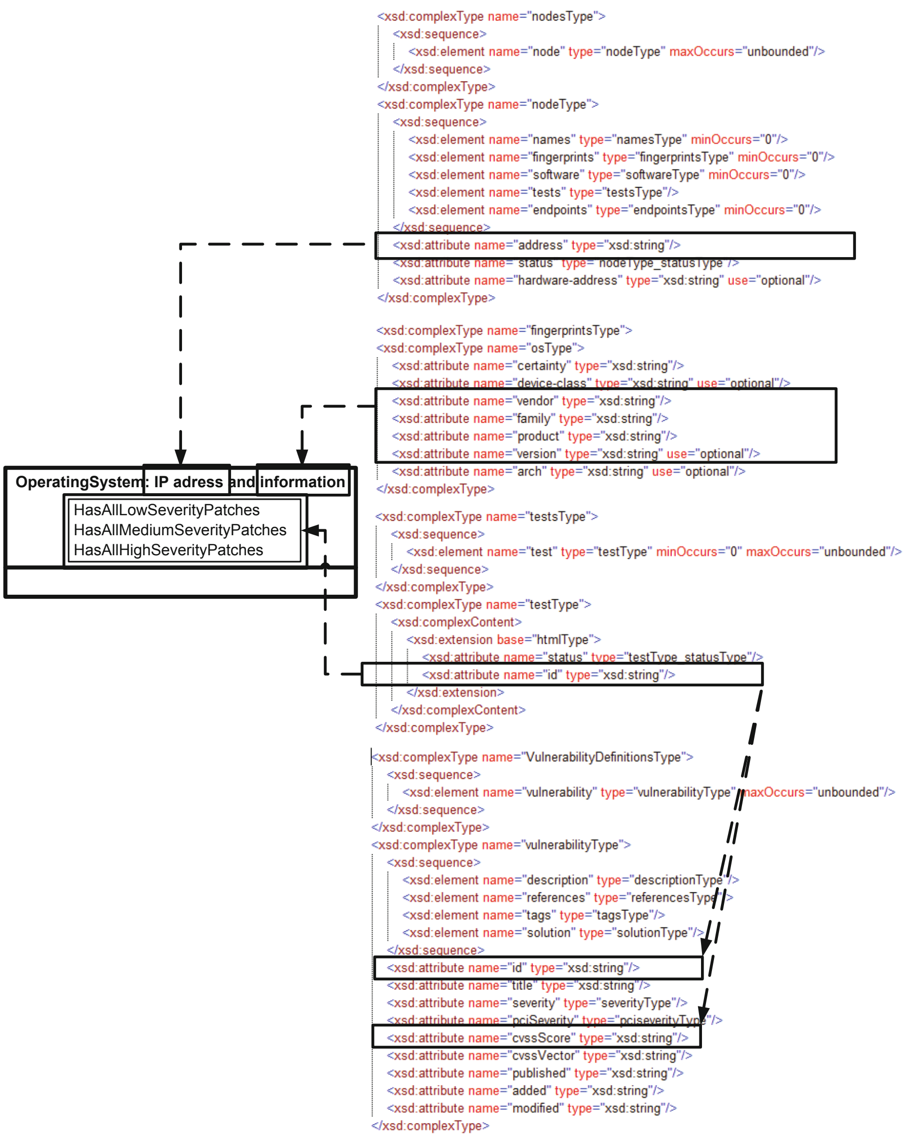
It is also possible to gather data for three of the attributes for the OperatingSystem class: Whether the operating system has any low, medium or high severity vulnerabilities as defined by the Common Vulnerability Scoring System [13] (**HasAll(Low/Medium/High)SeverityPatches**). The CVSS is a quantitative metric on a scale from 0.0-10.0 which grades IT security vulnerabilities according to their criticality. A low severity vulnerability corresponds to 0.0-3.9, a medium severity vulnerability to 4.0-6.9, and a high severity vulnerability to 7.0-10.0. NeXpose detects any vulnerabilities present in the probed operating system(s) and presents them with their corresponding CVSS score.

4.2 Software Service

The CySeMoL entity Service and three of its attributes are possible to automatically instantiate (cf. Figure 4. These concepts are described below. As can be seen, all information required by CySeMoL regarding these aspects are fulfilled.

Service. *CySeMoL*: A software that functions to service software clients. A Service listens on one or several ports for requests by software clients *NeXpose*: All relevant overall aspects can be captured, e.g. port number, software name and software version.

It is, as for the OperatingSystem class, also possible to gather data for three of the attributes of the Service class: Whether the service has any low, medium or high severity vulnerabilities (**HasAll(Low/Medium/High)SeverityPatches**) as defined by the CVSS. NeXpose detects any vulnerabilities present in the probed software service(s) and presents them with their corresponding CVSS score.



OperatingSystem: IP adress and information
HasAllLowSeverityPatches
HasAllMediumSeverityPatches
HasAllHighSeverityPatches

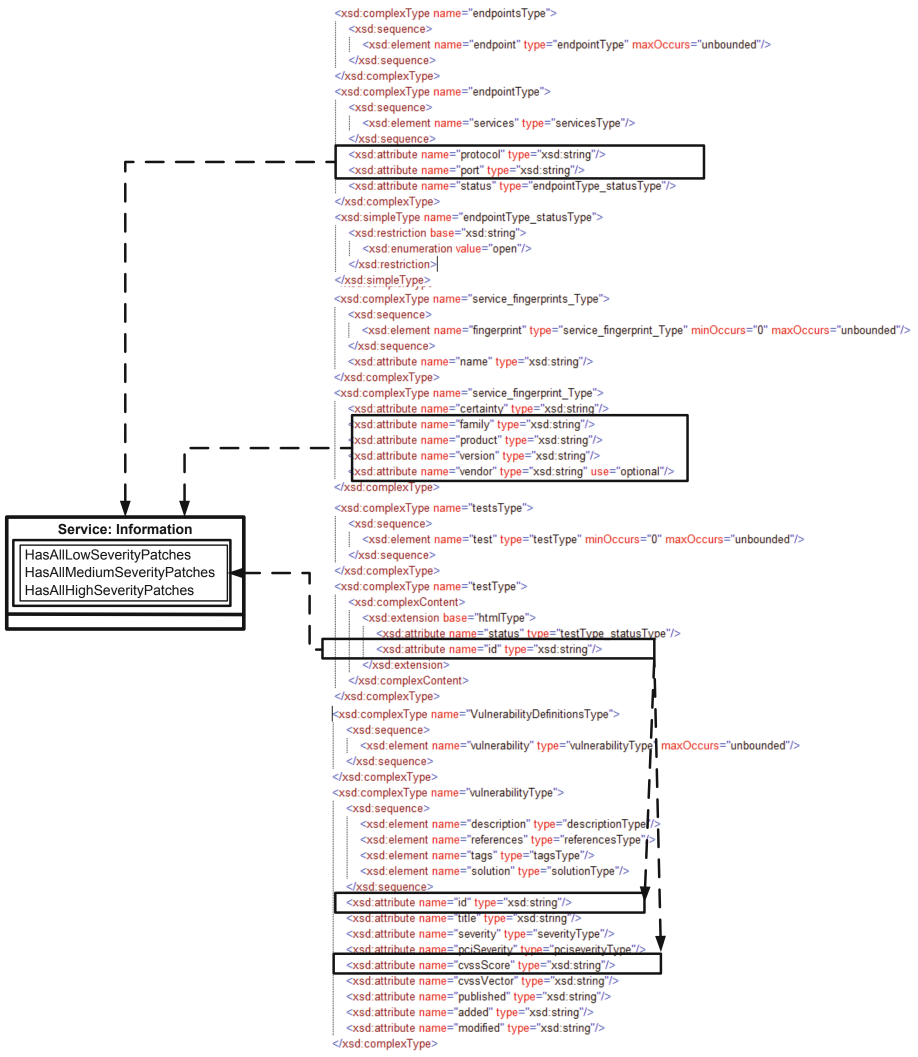
```
<xsd:complexType name="nodesType">
  <xsd:sequence>
    <xsd:element name="node" type="nodeType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="nodeType">
  <xsd:sequence>
    <xsd:element name="names" type="namesType" minOccurs="0"/>
    <xsd:element name="fingerprints" type="fingerprintsType" minOccurs="0"/>
    <xsd:element name="software" type="softwareType" minOccurs="0"/>
    <xsd:element name="tests" type="testsType"/>
    <xsd:element name="endpoints" type="endpointsType" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="address" type="xsd:string"/>
  <xsd:attribute name="status" type="nodeType_statusType"/>
  <xsd:attribute name="hardware-address" type="xsd:string" use="optional"/>
</xsd:complexType>

<xsd:complexType name="fingerprintsType">
<xsd:complexType name="osType">
  <xsd:attribute name="certainty" type="xsd:string"/>
  <xsd:attribute name="device-class" type="xsd:string" use="optional"/>
  <xsd:attribute name="vendor" type="xsd:string"/>
  <xsd:attribute name="family" type="xsd:string"/>
  <xsd:attribute name="product" type="xsd:string"/>
  <xsd:attribute name="version" type="xsd:string" use="optional"/>
  <xsd:attribute name="arch" type="xsd:string" use="optional"/>
</xsd:complexType>

<xsd:complexType name="testsType">
  <xsd:sequence>
    <xsd:element name="test" type="testType" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="testType">
  <xsd:complexContent>
    <xsd:extension base="htmlType">
      <xsd:attribute name="status" type="testType_statusType"/>
      <xsd:attribute name="id" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="VulnerabilityDefinitionsType">
  <xsd:sequence>
    <xsd:element name="vulnerability" type="vulnerabilityType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="vulnerabilityType">
  <xsd:sequence>
    <xsd:element name="description" type="descriptionType"/>
    <xsd:element name="references" type="referencesType"/>
    <xsd:element name="tags" type="tagsType"/>
    <xsd:element name="solution" type="solutionType"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string"/>
  <xsd:attribute name="title" type="xsd:string"/>
  <xsd:attribute name="severity" type="severityType"/>
  <xsd:attribute name="pciSeverity" type="pciseverityType"/>
  <xsd:attribute name="cvssScore" type="xsd:string"/>
  <xsd:attribute name="cvssVector" type="xsd:string"/>
  <xsd:attribute name="published" type="xsd:string"/>
  <xsd:attribute name="added" type="xsd:string"/>
  <xsd:attribute name="modified" type="xsd:string"/>
</xsd:complexType>
```

Fig. 3. Mapping of OperatingSystem to NeXpose XML

4.3 Network Zone

The CySeMoL entity NetworkZone was possible to map to the output of NeXpose (cf. Figure 5). This is detailed below.

NetworkZone. *CySeMoL*: A network which enables connected systems to communicate. *NeXpose*: The IP of each scanned node. As such, it is possible to



```
<xsd:complexType name="endpointsType">
  <xsd:sequence>
    <xsd:element name="endpoint" type="endpointType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="endpointType">
  <xsd:sequence>
    <xsd:element name="services" type="servicesType"/>
  </xsd:sequence>
  <xsd:attribute name="protocol" type="xsd:string"/>
  <xsd:attribute name="port" type="xsd:string"/>
  <xsd:attribute name="status" type="endpointType_statusType"/>
</xsd:complexType>
<xsd:simpleType name="endpointType_statusType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="open"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="service_fingerprints_Type">
  <xsd:sequence>
    <xsd:element name="fingerprint" type="service_fingerprint_Type" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="service_fingerprint_Type">
  <xsd:attribute name="certainty" type="xsd:string"/>
  <xsd:attribute name="family" type="xsd:string"/>
  <xsd:attribute name="product" type="xsd:string"/>
  <xsd:attribute name="version" type="xsd:string"/>
  <xsd:attribute name="vendor" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:complexType name="testsType">
  <xsd:sequence>
    <xsd:element name="test" type="testType" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="testType">
  <xsd:complexContent>
    <xsd:extension base="htmlType">
      <xsd:attribute name="status" type="testType_statusType"/>
      <xsd:attribute name="id" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="VulnerabilityDefinitionsType">
  <xsd:sequence>
    <xsd:element name="vulnerability" type="vulnerabilityType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="vulnerabilityType">
  <xsd:sequence>
    <xsd:element name="description" type="descriptionType"/>
    <xsd:element name="references" type="referencesType"/>
    <xsd:element name="tags" type="tagsType"/>
    <xsd:element name="solution" type="solutionType"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string"/>
  <xsd:attribute name="title" type="xsd:string"/>
  <xsd:attribute name="severity" type="severityType"/>
  <xsd:attribute name="pciSeverity" type="pciseverityType"/>
  <xsd:attribute name="cvssScore" type="xsd:string"/>
  <xsd:attribute name="cvssVector" type="xsd:string"/>
  <xsd:attribute name="published" type="xsd:string"/>
  <xsd:attribute name="added" type="xsd:string"/>
  <xsd:attribute name="modified" type="xsd:string"/>
</xsd:complexType>
```

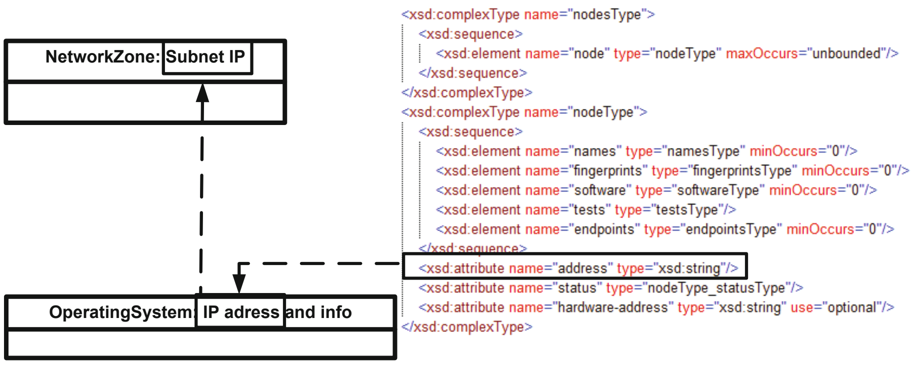
Fig. 4. Mapping of Service to NeXpose XML

group all systems (OperatingSystems) that have IPs on the same subnet (e.g. 172.168.2.x).

4.4 Software Product

The CySeMoL entity SoftwareProduct was possible to map to the output of NeXpose (cf. Figure 6). This is described below.

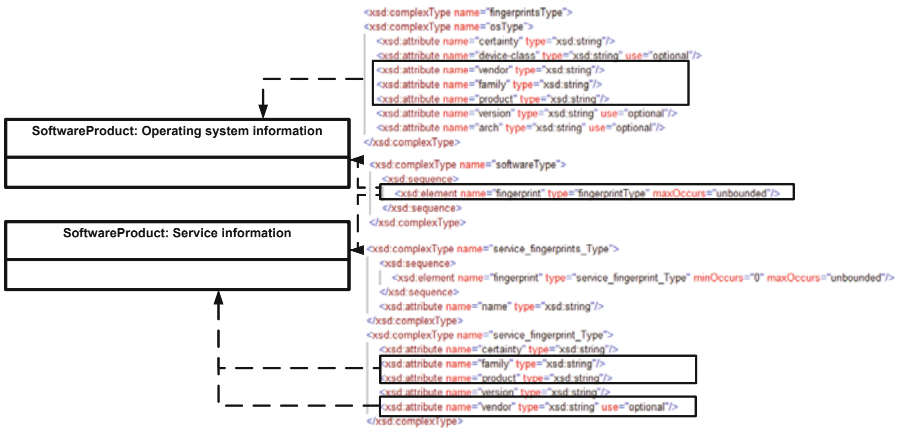
SoftwareProduct. *CySeMoL*: The type of software. For example, Windows XP SP2 or Apache Webserver. This is also the main difference from a Software-



```
<xsd:complexType name="nodesType">
  <xsd:sequence>
    <xsd:element name="node" type="nodeType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="nodeType">
  <xsd:sequence>
    <xsd:element name="names" type="namesType" minOccurs="0"/>
    <xsd:element name="fingerprints" type="fingerprintsType" minOccurs="0"/>
    <xsd:element name="software" type="softwareType" minOccurs="0"/>
    <xsd:element name="tests" type="testsType"/>
    <xsd:element name="endpoints" type="endpointsType" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="address" type="xsd:string"/>
  <xsd:attribute name="status" type="nodeType_statusType"/>
  <xsd:attribute name="hardware-address" type="xsd:string" use="optional"/>
</xsd:complexType>
```

Fig. 5. Mapping of NetworkZone to NeXpose XML

Installation (e.g. Service or OS) – a common computer network could include 30 systems which all use Windows XP SP2, but with different security patches installed. This would in CySeMoL correspond 30 instances of the entity OperatingSystem, all related to a single SoftwareProduct (Windows XP SP2). *NeXpose*: All general information needed by CySeMoL to generate the entity.


SoftwareProduct: Operating system information
SoftwareProduct: Service information

```
<xsd:complexType name="fingerprintsType">
<xsd:complexType name="osType">
  <xsd:attribute name="certainty" type="xsd:string"/>
  <xsd:attribute name="device-class" type="xsd:string" use="optional"/>
  <xsd:attribute name="vendor" type="xsd:string"/>
  <xsd:attribute name="family" type="xsd:string"/>
  <xsd:attribute name="product" type="xsd:string"/>
  <xsd:attribute name="version" type="xsd:string" use="optional"/>
  <xsd:attribute name="arch" type="xsd:string" use="optional"/>
</xsd:complexType>
<xsd:complexType name="softwareType">
  <xsd:sequence>
    <xsd:element name="fingerprint" type="fingerprintType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="service_fingerprints_Type">
  <xsd:sequence>
    <xsd:element name="fingerprint" type="service_fingerprint_Type" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="service_fingerprint_Type">
  <xsd:attribute name="certainty" type="xsd:string"/>
  <xsd:attribute name="family" type="xsd:string"/>
  <xsd:attribute name="product" type="xsd:string"/>
  <xsd:attribute name="version" type="xsd:string"/>
  <xsd:attribute name="vendor" type="xsd:string" use="optional"/>
</xsd:complexType>
```

Fig. 6. Mapping of SoftwareProduct to NeXpose XML

5 Example

In this section we describe how we tested the implementation on a real network. We give a brief introduction to the background of the collected data. Afterwards we depict how the resulting auto-generated model looks like.

5.1 The Setup

The main experimental setup was designed by the Swedish Defence Research Agency (FOI), with the support of the Swedish National Defence College (SNDC). Also, a group of computer security specialists and computer security researchers originating from various European governments, military, private sectors and academic institutions were part of designing the network architecture.

The environment was set to describe a simplified critical information infrastructure at a small electrical power utility. The environment was composed of 20 physical PC servers running a total of 28 virtual machines, divided into four VLAN segments. Various operating systems and versions thereof were used in the network, e.g. Windows XP SP2, Debian 5.0 and Windows Server 2003 SP1. Each host had several different network services operating, e.g. web-, mail-, media-, remote connection- and file sharing services. Furthermore, every host was more or less vulnerable through software flaws and/or poor configurations.

5.2 The Result

We performed an authenticated NeXpose scan on the setup environment and thereafter applied the mapping presented in chapter 4. The auto generated model consists of four instances of CySeMol's *NetworkZone* and 28 instances of CySeMol's *OperatingSystem* class. Furthermore 225 instances of the *Service* class and 141 instantiations of the *SoftwareProduct* class were automatically generated. The components were related based on the relations of CySeMol.

Figure 7, 8, and 9 show subsets of the resulting model, exemplary for one computer of the environment as the full model is too large to be shown here. In figure 7 the network zone that the computer belongs to is shown, followed by a consideration of the software and services that can be found on that machine. Finally figure 9 shows the computer and some of its attributes according to CySeMol, as well as the evidence that was set automatically.

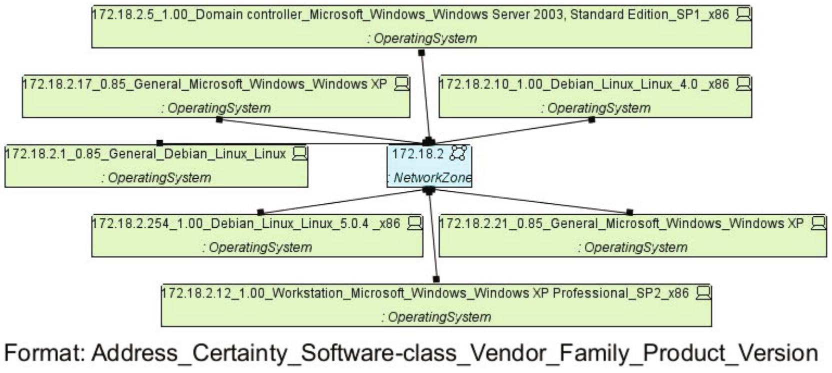

172.18.2.5_1.00_Domain controller_Microsoft_Windows_Windows Server 2003, Standard Edition_SP1_x86
: OperatingSystem
172.18.2.17_0.85_General_Microsoft_Windows_Windows XP
: OperatingSystem
172.18.2.10_1.00_Debian_Linux_Linux_4.0 _x86
: OperatingSystem
172.18.2.1_0.85_General_Debian_Linux_Linux
: OperatingSystem
172.18.2
: NetworkZone
172.18.2.254_1.00_Debian_Linux_Linux_5.0.4 _x86
: OperatingSystem
172.18.2.21_0.85_General_Microsoft_Windows_Windows XP
: OperatingSystem
172.18.2.12_1.00_Workstation_Microsoft_Windows_Windows XP Professional_SP2_x86
: OperatingSystem
Format: Address_Certainty_Software-class_Vendor_Family_Product_Version

Fig. 7. Resulting model of one network zone

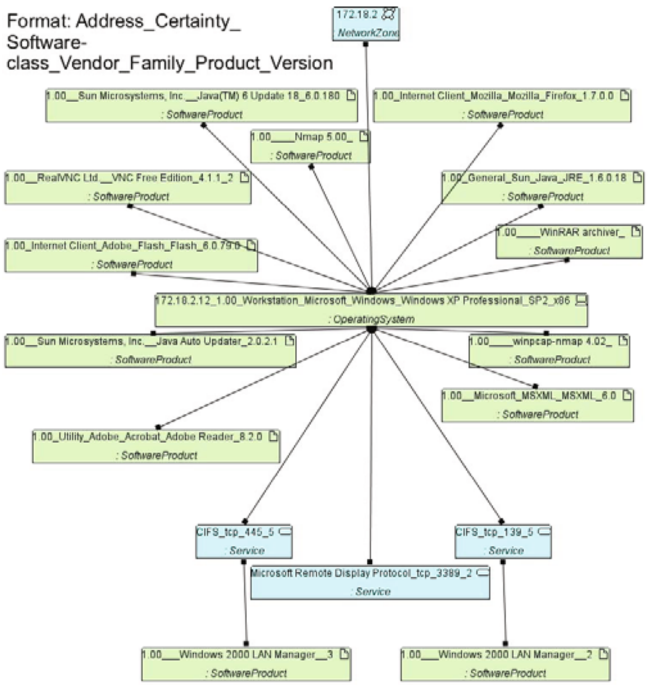

Format: Address_Certainty_Software-class_Vendor_Family_Product_Version
172.18.2 : NetworkZone
1.00__Sun Microsystems, Inc__Java(TM) 6 Update 18_6.0.180 : SoftwareProduct
1.00_Internet Client_Mozilla_Mozilla_Firefox_1.7.0.0 : SoftwareProduct
1.00___Nmap 5.00_ : SoftwareProduct
1.00__RealVNC Ltd__VNC Free Edition_4.1.1_2 : SoftwareProduct
1.00_General_Sun_Java_JRE_1.6.0.18 : SoftwareProduct
1.00___WinRAR archiver_ : SoftwareProduct
1.00_Internet Client_Adobe_Flash_Flash_6.0.79.0 : SoftwareProduct
172.18.2.12_1.00_Workstation_Microsoft_Windows_Windows XP Professional_SP2_x86 : OperatingSystem
1.00__Sun Microsystems, Inc.__Java Auto Updater_2.0.2.1 : SoftwareProduct
1.00___winpcap-nmap 4.02_ : SoftwareProduct
1.00__Microsoft_MSXML_MSXML_6.0 : SoftwareProduct
1.00_Utility_Adobe_Acrobat_Adobe Reader_8.2.0 : SoftwareProduct
CIFS_tcp_445_5 : Service
CIFS_tcp_139_5 : Service
Microsoft Remote Display Protocol_tcp_3389_2 : Service
1.00___Windows 2000 LAN Manager__3 : SoftwareProduct
1.00___Windows 2000 LAN Manager__2 : SoftwareProduct

Fig. 8. One selected node

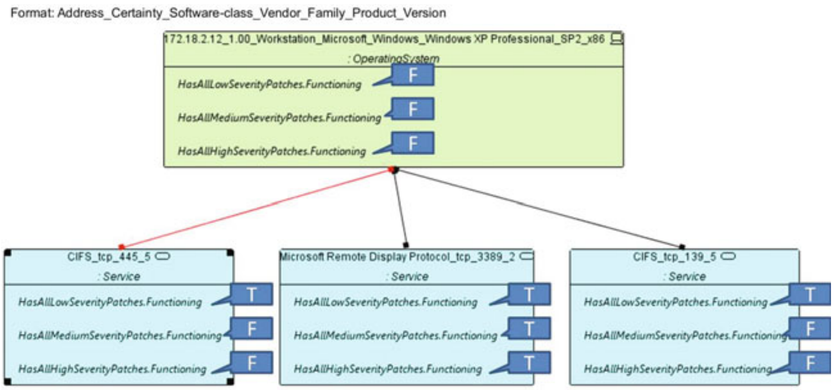

Format: Address_Certainty_Software-class_Vendor_Family_Product_Version
172.18.2.12_1.00_Workstation_Microsoft_Windows_Windows XP Professional_SP2_x86 : OperatingSystem
HasAllLowSeverityPatches.Functioning F
HasAllMediumSeverityPatches.Functioning F
HasAllHighSeverityPatches.Functioning F
CIFS_tcp_445_5 : Service
HasAllLowSeverityPatches.Functioning T
HasAllMediumSeverityPatches.Functioning F
HasAllHighSeverityPatches.Functioning F
Microsoft Remote Display Protocol_tcp_3389_2 : Service
HasAllLowSeverityPatches.Functioning T
HasAllMediumSeverityPatches.Functioning T
HasAllHighSeverityPatches.Functioning T
CIFS_tcp_139_5 : Service
HasAllLowSeverityPatches.Functioning T
HasAllMediumSeverityPatches.Functioning F
HasAllHighSeverityPatches.Functioning F

Fig. 9. The considered attributes of the node

6 Discussion and Future Work

This paper shows that vulnerability scanners can provide support for the creation of EA models. As mentioned earlier, scanners do not deliver complete EA models, but require some completion work. Their application however significantly reduces modeling effort and creates a model stub that can be complemented.

The validity and reliability of the approach can be discussed from two different viewpoints: i) how much of the meta-model that can be captured, both in scope (i.e. how much of the meta-model that can be instantiated) and context (i.e. if the scanner provides all the information needed to accurately capture the context of a variable), and ii) how accurate a vulnerability scanner is at assessing the instantiated variables. Regarding i), most of the more modeling intensive concepts of CySeMoL are captured and all context are accurate. That is, the scanner provides e.g. all the information regarding vulnerabilities that CySeMoL requires. Regarding ii), the scanning accuracy in terms of assessing vulnerabilities is studied in [10]. The accuracy in terms of assessing software, operating systems and such is something that will be examined in future works.

It would also be interesting to look at other variables provided by automated vulnerability scanning, e.g. user accounts of systems and software clients (such as Adobe Reader). Furthermore, automated scanning could be mapped to more commonly used EA frameworks such as [12] to increase the usage of the method.

Additionally in future work it might be investigated how other data sources can be used to further reduce the manual tasks. Examples of such sources are access control lists, ERP systems, and UDDI registries. Especially how automatic data collection for the domains that so far not have been considered (the Business Layer and the information architecture) can be done, needs to be investigated. The goal is to minimize the manual effort to generate EA models.

Enterprises are changing permanently. Periodic scans leading to an automatic model update might therefore be implemented in the present tool as well.

It is also possible to collect information on vulnerabilities of services and software. This is something that we aim to incorporate in a future project.

7 Conclusion

In this paper we presented an extension of our previously developed tool that allows the automatic generation of elements for EA models. The input is provided by a vulnerability scanner, which was used to identify elements that were part of a computer network. Our implementation is generic even though CySeMol, a meta-model for security analysis, was used as a running example. The data gained from the vulnerability scanner can be used to instantiate any meta-model, once a mapping has been defined. The scan with NeXpose took less than an hour and the creation of the EA model using that data was next to instantaneous. Thus, it should be a viable option for EA architects. We have also illustrated the architecture of our implementation and described used components in detail. Finally, we have presented a practical application based on real data of our implementation. Thereby we have shown the feasibility of our approach.

References

1. 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)

2. Aier, S., Kurpjuweit, S., Saat, J., Winter, R.: Enterprise Architecture Design as an Engineering Discipline. AIS Transactions on Enterprise Systems 1(1), 36–43 (2009)
3. BiZZdesign: BiZZdesign Architect (March 2011) http://www.bizzdesign.com
4. Breu, R.: Ten principles for living models - a manifesto of change-driven software engineering. In: International Conference on Complex, Intelligent and Software Intensive Systems, pp. 1–8 (2010)
5. Buckl, S., Matthes, F., Neubert, C., Schweda, C.M.: A wiki-based approach to enterprise architecture documentation and analysis. In: 17th European Conference on Information Systems, pp. 1–13 (2009)
6. Buschle, M., Ullberg, J., Franke, U., Lagerström, R., Sommestad, T.: A Tool for Enterprise Architecture Analysis Using the PRM Formalism. In: Soffer, P., Proper, E. (eds.) CAiSE Forum 2010. LNBIP, vol. 72, pp. 108–121. Springer, Heidelberg (2011)
7. Fischer, R., Aier, S., Winter, R.: A federated approach to enterprise architecture model maintenance. Enterprise Modelling and Information Systems Architectures 2(2), 14–22 (2007)
8. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: Proc. of the 16th International Joint Conference on Artificial Intelligence, pp. 1300–1309. Morgan Kaufmann (1999)
9. Hafner, M., Winter, R.: Processes for enterprise application architecture management. In: Proceedings of the 41st Hawaii International Conference on System Sciences, pp. 396–406 (2008)
10. Holm, H., Sommestad, T., Almroth, J., Persson, M.: A quantitative evaluation of vulnerability scanning. Information Management & Computer Security 19(4), 231–247 (2011)
11. Johnson, C., Quinn, S., Scarfone, K., Waltermire, D.: The technical specification for the security content automation protocol (SCAP). NIST Special Publication 800, 126 (2009)
12. Lankhorst, M.M.: Enterprise Architecture at Work: Modelling, Communication and Analysis, 2nd edn. Springer, Heidelberg (2009)
13. Mell, P., Scarfone, K., Romanosky, S.: A complete guide to the common vulnerability scoring system version 2.0. In: Published by FIRST-Forum of Incident Response and Security Teams (2007)
14. Rapid7: NeXpose (March 2011), http://www.rapid7.com/
15. Software AG: ARIS for SAP (2011), http://www.softwareag.com/corporate/products/aris_platform/aris_implementation/aris_sap
16. Sommestad, T., Ekstedt, M., Johnson, P.: A probabilistic relational model for security risk analysis. Computers & Security 29(6), 659–679 (2010)
17. Sommestad, T., Ekstedt, M., Nordström, L.: A case study applying the Cyber Security Modeling Language (2010)
18. Sousa, P., Lima, J., Sampaio, A., Pereira, C.: An Approach for Creating and Managing Enterprise Blueprints: A Case for IT Blueprints. In: Albani, A., Barjis, J., Dietz, J.L.G. (eds.) CIAO! 2009. LNBIP, vol. 34, pp. 70–84. Springer, Heidelberg (2009)
19. The Open Group: The Open Group Architecture Framework (TOGAF) - version 9. The Open Group (2009)
20. Troux Technologies: Metis (March 2011), http://www.troux.com/products/
21. Zachman, J.A.: A framework for information systems architecture. IBM Systems Journal 26, 276–292 (1987)

Creating Declarative Process Models Using Test Driven Modeling Suite

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 promise a high degree of flexibility. However, current declarative state-of-the-art modeling notations are, while sound on a technical level, hard to understand. To cater for this problem, in particular to improve the understandability of declarative process models as well as the communication between domain experts and model builders, Test Driven Modeling (TDM) has been proposed. In this tool paper we introduce Test Driven Modeling Suite (TDMS) which provides operational support for TDM. We show how TDMS realizes the concepts of TDM and how Cheetah Experimental Platform is used to make TDMS amenable for effective empirical research. Finally, we provide a brief example to illustrate how the adoption of TDMS brings out the intended positive effects of TDM for the creation of declarative process models.

Keywords: Declarative Business Process Models, Test Driven Modeling, Test Driven Modeling Suite.

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]. 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 [2]. 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 [3], case handling [4], declarative processes [5] and late binding and modeling [6] (an overview is provided in [7]).

Although declarative processes promise a high degree of flexibility, avoid overspecification and provide more maneuvering for end-users [8], [5], they are not widely adopted in practice yet. In particular, as pointed out in [8], [9], [10], understandability problems hamper the usage of declarative process models. For instance, checking whether a process instance is supported by a process schema, is far from trivial. An approach tackling these problems, the *Test Driven*

Modeling (TDM) methodology, is presented in [10]. TDM aims at improving the understandability of declarative process models as well as the communication between domain experts [11] and model builders [11] by adopting the concept of *test cases* from software engineering. The contribution of this paper is to describe *Test Driven Modeling Suite* (TDMS)[1], i.e., the software that provides operational support for TDM.

The remainder of this paper is structured as follows: Section 2 briefly introduces declarative business process models, Section 3 shortly discusses TDM. Then, Section 4 describes the software architecture and features of TDMS, whereas Section 5 illustrates the usage of TDMS by an example. Finally, Section 6 deals with related work and Section 7 concludes with a summary and an outlook.

2 Declarative Process Models

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 [8]. 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 [8]).

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). *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. This is indicated by the 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. This is indicated

[1] Freely available from: `http://www.zugal.info/tdms`

by the 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). Hence, 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.

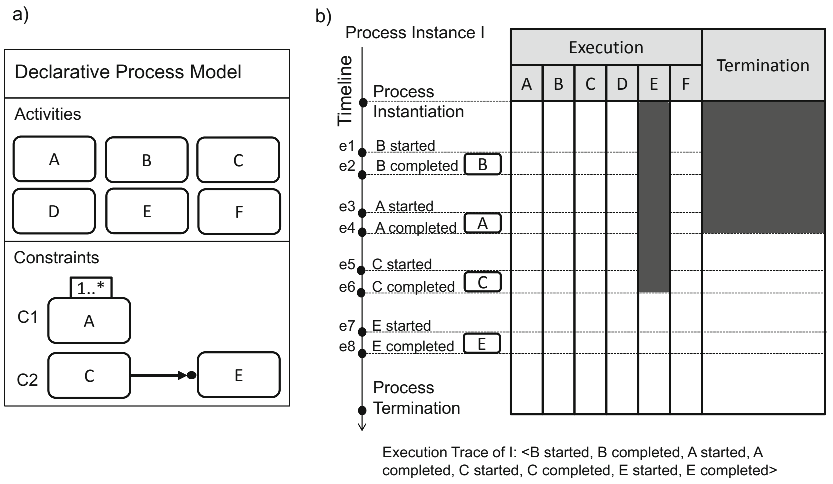


Fig. 1. Executing a declarative process

As illustrated in Fig. 1 b), 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 end events, e.g., <*B started*, *B completed*, *A started*, *A completed*> is abbreviated by <*B*, *A*>.

3 Test Driven Modeling

In the following, we briefly introduce TDM, the conceptual basis of TDMS. In particular, in Section 3.1 we will draw on concepts from cognitive psychology to shed light on possible causes of understandability problems related to declarative models. Subsequently, in Section 3.2, we discuss the most relevant concepts of TDM.

3.1 Cognitive Backgrounds

Declarative process models allow for the specification of flexible business processes [8], [12]. Still, as argued in [8], [9], [10], the understandability of respective models appears to be a hurdle for practical usage. Understandability thereby refers to how difficult it is to extract information from a process model. As detailed in [13], understandability is usually operationalized by asking questions about a process model. The more questions are answered correctly on average, the higher the understandability. Currently, it is still unclear for which reasons declarative models are harder to understand than imperative models. To provide a possible explanation, we would like build upon concepts from cognitive psychology. In particular, we identified that *computational offloading* [14], [15], [16] seems to play an essential role. In short, computational offloading allows the reader to *"offload"* computations to a diagram. In other words, the way how the diagram represents information allows the reader to quickly extract certain information. For instance, in a BPMN model control flow is explicitly represented by sequence flows (i.e., control edges) and gateways (e.g., AND gateway, OR gateway). Assume the reader wants to check whether a certain process instance is supported by an imperative process model. To this end, she may use the control edges to simulate the process instance by tracing through the process model. In this way, the model allows to offload the computation of the process instance. Contrariwise, one might describe the process model textually. Both representations (text and diagram) are information equivalent, i.e., the same information is present, however, the text does not allow the reader to quickly identify process instances, the reader has to simulate the process instance entirely in her head. Similarly, declarative process models do not provide explicit mechanisms to offload the computation of execution traces. Rather, as discussed in Section 2, the reader has to interpret the constraints in her mind. For a detailed discussion about computational offloading and related cognitive concepts, we refer to [17].

Assuming that computational offloading of computing process instances is not present in declarative models, i.e., reading imposes a high mental effort, repercussions on understandability, validation and maintainability can be expected. Since model understandability, as defined in [18], directly relates to reading and answering questions about a process model, as discussed above, a negative impact can be expected. Regarding validation, i.e., to check whether the model properly reflects the real-world business, an interesting insight is provided in [19]. The authors state that *"programmers rely heavily upon mental simulation for evaluating the validity of rules"*. Seen in the context of business process modeling, "mental simulation" refers to the "mental execution" of process instances. In other words, the person who validates the process model checks via "mental simulation" whether certain process instances, i.e., scenarios, are supported by a process model. As discussed, for a declarative business process model, the computation of process instances is far from trivial, hence a negative impact on validation can be expected. In further consequence, also the maintainability of declarative process models may be compromised, as argued in [10]. It is known that every change operation requires a *sense-making task*, i.e., determining what to change, as well as an *action task*, i.e., performing the change [20]. Compromised understandability supposedly compromises the sense-making task, which in turn impairs the change operation.

Basically, the idea of TDM is to provide computer-based support to compensate for the lack of computational offloading. In particular, *test cases* allow to capture and automatically validate scenarios, i.e., process instances, that should be supported by the process model. Likewise, test cases provide the option to specify anti-scenarios, i.e., behavior that should be forbidden by the process model.

3.2 Test Driven Modeling

Constraints, as introduced in Section 2, focus on forbidden behavior. TDM, however, introduces the concept of *test cases* to test for *desired* behavior of the process model. In particular, as illustrated in Fig. 2, TDM's meta model can be divided into two main parts: the specification of test cases (upper half) as well as the specification of the business process model (lower half). A *Test Driven Model* consists of exactly one *Declarative Process Model* and an arbitrary amount of *Test Cases*. A *Declarative Process Model*, as already discussed in Section 2, consists of at least one *Activity* as well as an arbitrary amount of *Constraints*. For the sake of brevity, Fig. 2 shows three constraints only, i.e., the *Response Constraint*, the *Precedence Constraint* and the *Coexistence Constraint*. TDMS actually supports all constraints described in [8], for a detailed description of the constraints we refer to [12]. Besides the specification of a *Declarative Process Model*, the meta model in Fig. 2 describes how *Test Cases* can be specified. In particular, a *Test Case* is built-up by a *Process Instance* (subsequently also referred to as *execution trace*) and an arbitrary number of *Assertions*. The *Process Instance*, in turn, consists of an arbitrary number of *Activity Instances*. For each *Activity Instance* a start event (i.e., when the activity instance enters the state *started* in its life-cycle) as well as and an end event (i.e., when the activity instance is *completed*) are defined. Similarly, each *Assertion* is defined for a certain window by its start- and end event. Within this window, a condition that is specified by the *Assertion* must hold. TDM thereby differentiates between two types of *Assertions*: an *Execution Assertion* can be used to verify whether a certain *Activity* is executable. The *positive* flag in *Assertion* thereby defines whether an *Activity* is expected to be *executable* or if the *Activity* is expected to be *non-executable*. Similarly, a *Termination Assertion* can be used to test whether the *Process Instance* can be terminated within a specified window.

In short, TDM allows for the specification of declarative process models and test cases. Each test case defines a certain scenario, i.e., process instance, that must be supported by the process model. Assertions can thereby be used to test for specific conditions, namely whether an activity is executable as well as whether the process instance can be terminated.

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

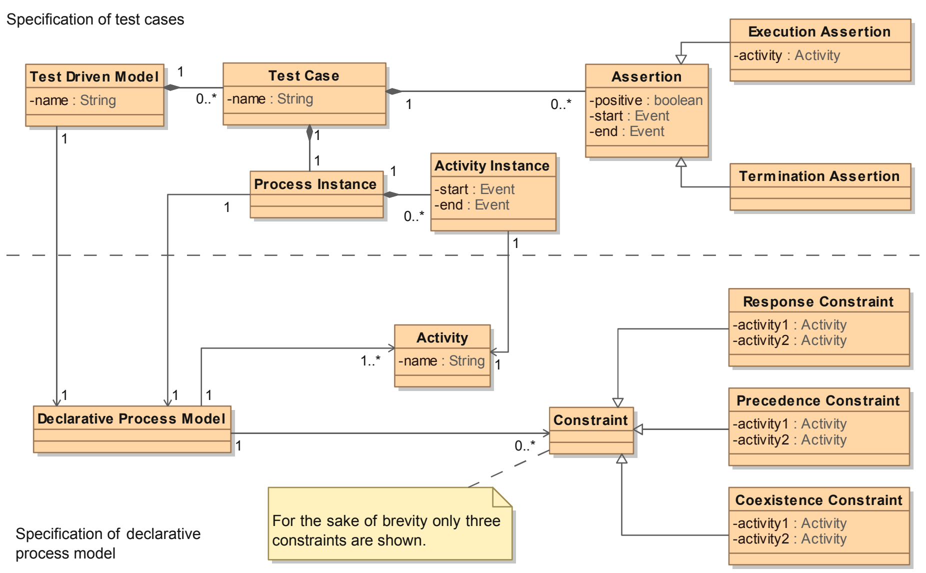


Fig. 2. Meta model of TDM

instance cannot be terminated before *e2* (3), however, it must be possible to terminate after *e2* (4). For the reason of simplicity, the example shows subsequent executions only. However, testcases can also be used to simulate parallel executions of activity. In this vein, also several different instances of the same activity may run at the same time—given that no constraint prohibits such behavior. The times in Fig. 3 do not necessarily constitute *real* times, but rather provide a timeline to test for control-flow behavior, i.e., define whether activities can be executed subsequently or in parallel. Furthermore test cases are validated automatically, i.e., no user interaction is required to check whether the specified behavior is supported by the process model.

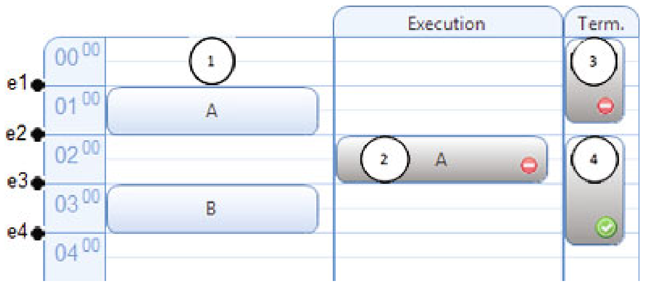


Fig. 3. A simple testcase

To illustrate how a test case may help to improve the understandability of a declarative process model, consider the test case illustrated in Fig. 4. The process model to the right (2) can be described in the following way: A must be executed exactly once (cf. cardinality constraint on A). After A has been executed, B must be executed (cf. response constraint between A and B). Thus, also B must be executed at least once for every process instance. However, this information is present in the process model implicitly only. Hence, the person who reads to model has to inspect the model carefully for such dependencies in order to properly understand the models' semantics—computational offloading is missing. According to [20], connections that are not directly visible in a model are referred to as *hidden dependencies*. As the name suggests, such dependencies are hard to see and detect, potentially misleading the reader and causing understandability problems. TDM allows the process modeler to actively resolve hidden dependencies by specifying a respective test case, thereby making the dependency explicit. To illustrate how this could be done for the given example, consider Fig. 4 (1): the test case specifies that the process instance can only be terminated if B has been executed at least once, making the hidden dependency explicit. As soon as the modeler conducts changes to the process model that violate the test case, the automated validation of TDMS (cf. Section 4) immediately informs the modeler, making her aware of the hidden dependency.

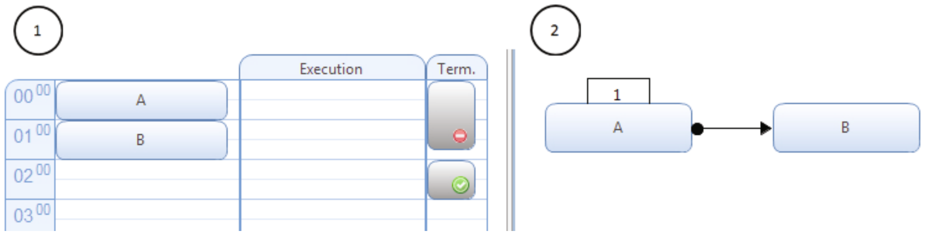


Fig. 4. Hidden dependency

So far we have introduced the concept of test cases and the intended impact on model understandability, in the following we will sketch how their adoption intends to improve the communication between domain expert (DE) and model builder (MB). First, it is worthwhile to note that test cases and process model are not meant to be created in isolation. Rather, as inspired by Test Driven Development [21], test cases and process model should be created interwoven (for a detailed discussion we refer to [10]). Thereby, test cases provide information in a form that is not only understandable to the MB, but also understandable to the DE, who normally does not have the knowledge to read formal process models [11]. Usually the DE needs the MB to retrieve information from the model, cf. Fig. 5 (2) and (3). Since test cases are understandable to the DE, they provide an additional communication channel to the process model, cf. Fig. 5 (4) and (6). It is important to stress that TDM's intention is not to make the DE specify the test cases in isolation. Rather, test cases should be created by the DE and the MB together and provide a common basis for discussion.

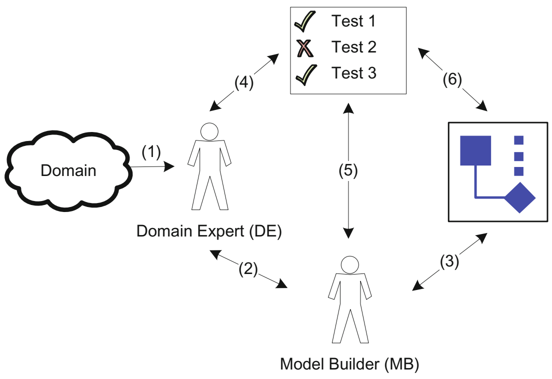


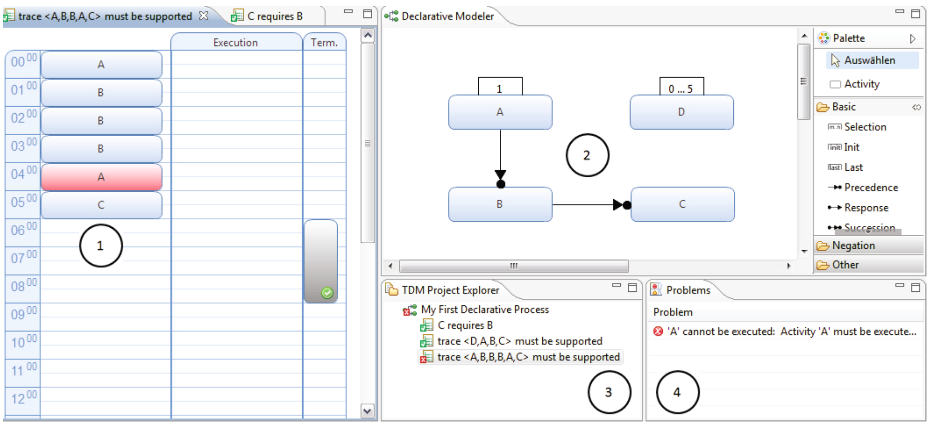
Fig. 5. Communication flow

4 Test Driven Modeling Suite

Up to now we have introduced the concept of TDM. This section deals with Test Driven Modeling Suite (TDMS) that provides operational support for TDM. In particular, Section 4.1 discusses the features of TDMS in detail. Subsequently, Section 4.2 describes how TDMS is integrated with existing frameworks for empirical research and business process execution.

4.1 Software Components

To give an overview of TDMS' features, a screenshot is provided in Fig. 6; each component will be described in detail in the following. On the left hand side TDMS offers a graphical editor for editing test cases (1). To the right, a graphical editor allows for designing the process model (2). Whenever changes are conducted, TDMS immediately validates the test cases against the process model and indicates failed test cases in the test case overview (3). In this case, it lists three test cases from which one failed. In addition, TDMS provides a detailed problem message about failed test cases in (4). In this example, the MB 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. In TDMS the failed test case is indicated by the activity highlighted in (1), the test cases marked in (3) and the detailed error message in (4).



My First Declarative Process
C requires B
trace <D,A,B,C> must be supported
trace <A,B,B,B,A,C> must be supported

Problem
'A' cannot be executed: Activity 'A' must be execute...

Fig. 6. Screenshot of TDMS

Testcase Editor. As discussed in Section 3, test cases are a central concept of TDM, have precise semantics for the specification of behavior and still should be understandable to domain experts. To this end, TDMS provides a calendar-like test case editor as shown in Fig. 6 (1). Whether the user interface is indeed as intuitive, i.e., self-explanatory, is not entirely clear yet. So far we know that a group of students was able to use it after a short introduction [22]. In addition, further investigations into its usability are planned, cf. Section 7.

Declarative Process Model Editor. The declarative process model editor, as shown in Fig. 6 (2), provides a graphical editor for designing models in DecSerFlow [8], a declarative process modeling language.

Testcase Creation and Validation. In order to create new test cases or to delete existing ones, Fig. 6 (3) provides an outline of all test cases. Whenever a test cases is created, edited or deleted, or, on the other hand, the process model is changed, TDMS immediately validates all test cases. For the case a test case fails, TDMS provides a detailed problem message in Fig. 6 (4). It is important to stress that the validation procedure is performed *automatically*, i.e., no user interaction is required to validate the test cases. To this end, TDMS provides a *test engine* in which test cases are executed, as shown in Fig. 7. Basically, the test engine consists of a declarative process instance that is executed on a declarative workflow engine within a test environment. Thereby, TDMS' process model provides the basis for the process instance. The test cases steer the execution of the process instance, e.g., instantiating the process instance, starting activities or completing activities. In addition, test cases may also check the state of the process instance in the course of evaluating execution- or termination assertions. For a detailed description of test case validation, we refer to [10].

As pointed out in Section 3, the TDM methodolog is iterative, hence TDMS must also provide respective support. In particular, the iterative creation of

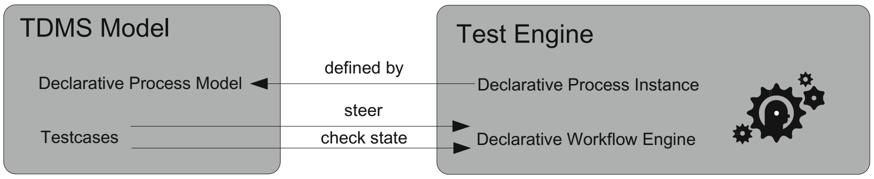

TDMS Model
Declarative Process Model
Testcases
defined by
steer
check state
Test Engine
Declarative Process Instance
Declarative Workflow Engine

Fig. 7. Testing framework

the process model poses a significant challenge, as any relevant change of the process model[2] requires the validation of testcases. However, existing declarative approaches either lead to exponential runtime for schema adaptations [8] or do not support workflow execution [12]. In order to tackle these problems, TDMS provides an own declarative workflow engine. Similar to Declare, where constraints are mapped to LTL formulas [23], TDMS' workflow engine maps constraints to Java[3] objects. In addition, for each process instance, the workflow engine keeps a list of events to describe its current state, as described in Section 2. The enablement of an activity can then be determined as detailed in the following. Based on the current process instance, a constraint is able to determine whether it restricts the execution of an activity. The workflow engine consults all defined constraints and determines for each constraint whether it restricts the execution. If no constraint vetos, the activity can be executed. For determining whether the process instance can be terminated, a similar strategy is followed. However, in this case constraints are asked whether they restrict the termination of the process instance instead.

Whenever a constraint should be added to the process model it is then sufficient to add this constraint to set of constraints to be checked. Similarly, when removing a constraint, the workflow engine does not consider the respective constraint anymore. While such an approach allows for efficient schema adaptations, it does not support verification mechanisms as provided in, e.g., Declare [23]. To compensate for this shortcoming, TDMS provides an interface to integrate third party tools for verification (cf. Section 4.2).

In order to ensure that all components work properly, TDMS has been developed using Test Driven Development [21], where applicable. In addition, researchers with different backgrounds, e.g., economics and computer science, have been included to develop an intuitive, i.e., self-explanatory, user interface. To validate whether our efforts succeeded, we used TDMS to teach declarative process modeling. In particular, we made use of TDMS' validation of test cases to allow students to interactively explore the semantics of a declarative process model. After a short introduction, students were able to work independently, indicating that operating TDMS, i.e., using the software, is easy to learn. Regarding the quality of TDMS, we would like to refer to a controlled experiment we recently

[2] Layouting operations, for instance, can be ignored here as they do not change the semantics of the process model.

[3] http://java.sun.com

performed [22]. Thereby, 12 students used TDMS for about 2 hours to adapt 2 declarative process models, i.e., a total of 24 process models were adapted. Throughout the experiment, no abnormal program behavior was observed. Apparently this does not mean that TDMS has industrial quality, however, TDMS meets the requirement for academic purposes as intended.

4.2 Integration of Test Driven Modeling Suite

TDM, as introduced in Section 3, focuses on the modeling of declarative processes, TDMS provides the necessary operational support, i.e., tool support. To this end, TDMS makes use of Cheetah Experimental Platform's (CEP) [24] components for empirical research and integrates Declare [23] for workflow execution and process model verification, as illustrated in Fig. 8 and detailed in the following.

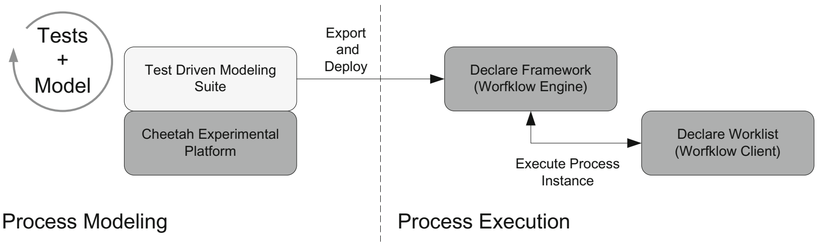

Tests
+
Model
Test Driven Modeling Suite
Cheetah Experimental Platform
Export and Deploy
Declare Framework (Worfklow Engine)
Declare Worklist (Worfklow Client)
Execute Process Instance
Process Modeling
Process Execution

Fig. 8. Interplay of TDMS, CEP and Declare

Cheetah Experimental Platform as Basis. One of the design goals of TDMS was to make it amenable for empirical research, i.e., it should be easy to employ in experiments. In addition, data should be easy to collect and analyze. For this purpose, TDMS was implemented as an experimental workflow activity of CEP, allowing TDMS to be integrated in any experimental workflow (i.e., a sequence of activities performed during an experiment, cf. [24]). Furthermore, we use CEP to instrument TDMS, i.e., to log each relevant user interaction to a central data storage. This logging mechanism, in combination with CEP's replay feature, allows the researcher to inspect in detail how TDMS is used to create process models and test cases step-by-step. Or, even more sophisticated, such a fine-grained instrumentation allows researchers and practioners to closely monitor the *process of process modeling*, i.e., the creation of the process model, using *Modeling Phase Diagrams* [25].

To illustrate how using CEP as basis for TDMS is beneficial for empirical research, we would like to refer to a recently performed experiment [22]. Therein, we investigated the impact of testcases on the maintainability of declarative process models. To this end, we provided students with two modeling assignments. For one of the modeling assignments, the full support of TDMS was available. For the other modeling assignment, only the process model editor was available.

In addition, we used a survey to assess demographic data such as modeling experience or education (for details we refer to [22]). The first benefit of CEP is that all these tasks are *automatically* presented to the students. Hence, no student could accidentally forget to fill out the demographic survey or to perform a modeling task. In other words, TDMS can seamlessly be integrated in such an experimental workflow. Thereby, all data is automatically collected and stored in a database. The second benefit of CEP comes out when evaluating the data gathered during the experiment. On the one hand, data can be exported *in an automated way* to comma-separated value files, which can then directly be analyzed using statistics software. On the other hand, collected data is fine-grained and therefore allows for in-depth evaluation. In this sense, we could show that with testcases at hand, twice as many constraints were added or deleted [22].

Process Model Verification and Execution. As discussed, the internal workflow engine of TDMS does not support the verification of declarative process models. However, it is known that the combination of constraints may lead to activities that can not be executed [8]. In order ensure that the process model is free from such *dead activities*, we make use of the verification provided in Declare [23]. In particular, as illustrated in Fig. 8, the process model is iteratively created in TDMS. For the purpose of verification, the process model is then converted into a format that can be read by the Declare framework. Similarly, this export mechanism can be used to execute the process model in Declare's workflow engine.

5 Example

A preliminary empirical evaluation shows a positive influence of TDM during model maintenance [22]. First, mental effort decreased, i.e., less cognitive resources were needed to conduct the change. Second, perceived quality increased, i.e., modelers were more confident about their changes—even though the quality of changes did not differ significantly. To illustrate the influence of TDMS on process modeling, we provide an example that shows how a DE and a MB could use TDMS to create a process model and respective test cases describing of how to supervise a master thesis (cf. Fig. 9–11). For the sake of brevity, the example is kept on an abstract level and the following abbreviations are used:

D: *Discuss topic* **P:** *Provide feedback* **G:** *Grade work*

Starting from an empty process model, the DE lines out general properties of the process: *"When supervising a master thesis, at first the topic needs to be discussed with the student. While the student works on his thesis, feedback may be provided at any time. Finally, the thesis needs to be graded."*. Thus, possibly with help of the MB, the DE inserts activities D, P and G in the test case's execution trace (cf. Fig. 9). TDMS automatically creates respective activities in the process model and the DE and MB run the test case. As the specified execution trace is supported by the process model, the test case passes.

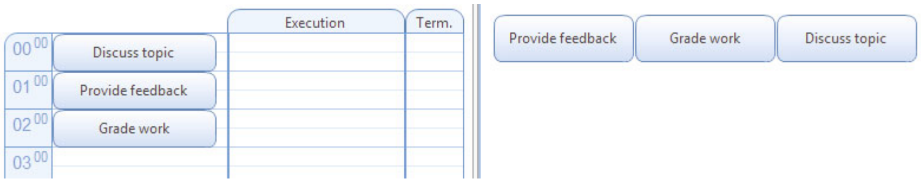


Fig. 9. Testcase 1: <*D*,*P*,*G*> proposed by the DE

Subsequently, the DE and MB engage in a dialogue of questioning and answering [26]—the MB challenges the model: *"So every thesis must start by discussing the topic?"*. *"Yes, indeed—you need to establish common knowledge first."*, the DE replies. Thus, they create a new test case capturing this requirement and run it. Apparently, the test case fails as there are no constraints in the model yet. The MB inserts an init constraints on D (i.e., D must be the first activity in every process instance); now the test case passes (cf. Fig. 10).

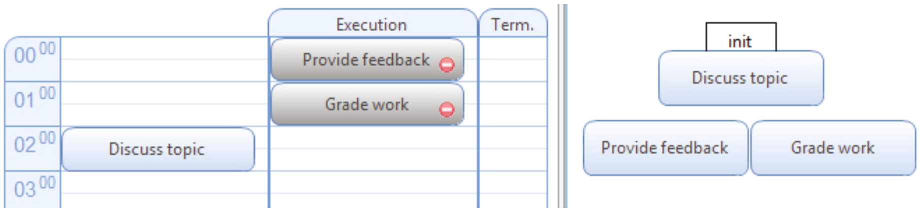


Fig. 10. Testcase 2: Introduction of Init on D

Again, the MB challenges the model and asks: *"Can the supervisor grade a thesis multiple times?"*. The DE replies: *"No, of course not, each thesis must be graded exactly once."* and together they specify a *third test case* that ensures that G must be executed exactly once. By automatically validating this test case, it becomes apparent that the current model allows G to be executed several times. Thus, the MB introduces a cardinality constraint on G (cf. Fig. 11).

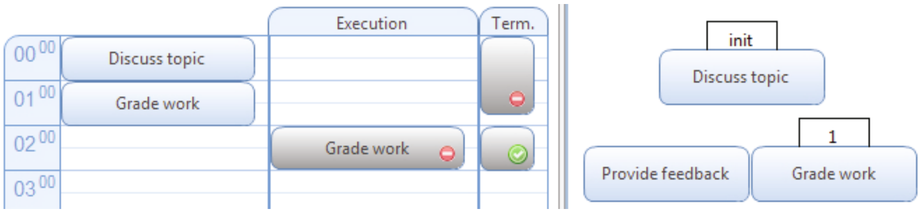


Fig. 11. Testcase 3: Introduction of Cardinality on G

While this example is kept small for the sake of brevity, it illustrates the benefits of using TDMS for modeling. First, the DE, who is usually not trained in reading or creating formal process models [11], is not required to modify the model itself, rather he defines behavior through the specification of test cases (possibly with the help of the MB). Second, test cases provide a common basis for understanding, thus supporting communication between the DE and MB. Third, behavior that is specified through test cases is validated automatically by TDMS, thereby ensuring that model changes do not violate desired behavior. In this sense, test cases can be seen as a computer-supported kind of *modeling minutes* [27] that can be automatically validated against the process model.

6 Related Work

TDMS, as described in this work, allows for the interweaved creation of test cases and process models. The combination of conceptual models and test cases is far from new. For instance, in [28], [29], a language supporting automated test cases for conceptual schemas is presented. In contrast to this work, the language is not designed to be understood by the DE. In so-called *scenario-based approaches* the goal is to synthesize a conceptual model from a set of *scenarios*, i.e., test cases. The main difference to this work is the way how models are created. In our work, it is the responsibility of the MB to create the model. In the approaches described in the following, the model is *automatically* synthesized from scenarios. This way of synthesizing models is applicable to a variety of modeling languages and domains, as shown in [30], [31]. For instance, scenarios may be specified in Message Sequence Charts, Sequence Diagrams, Collaboration Diagrams [31] or Petri nets [32], [33]. Scenarios can then be synthesized to, e.g., Statecharts, Automatons [31] or Petri nets [32]. In principle, such approaches may be also applied to declarative process modeling. In this vein, Lamma et al. [34] describe how to extract declarative process models from process logs. While automated synthesis of declarative process models is certainly a viable approach to follow, as pointed out in [35], it is questionable in how far models that have been synthesized automatically are readable.

7 Summary and Outlook

In this work we started by introducing declarative business process models and associated problems. In particular, we lined out how the lack of computational offloading as well as the presence of hidden dependencies compromises model understandability, validation and maintainability. Subsequently, we sketched the most important concepts of TDM and discussed how it intends to improve the understandability of declarative process models and supports the communication between DE and MB. Then, we described TDMS that provides operational support for TDM. Thereby, we sketched how we employ CEP as basis to make TDMS amenable for empirical research and showed how Declare is employed for the execution of declarative processes modeled in TDMS. Finally, we illustrated the intended usage of TDMS, in particular the iterative development of test cases and process model, with the help of a small example.

We acknowledge that it is not yet entirely clear whether TDM and TDMS in particular help to foster the communication between DE and MB as well as improve the understandability and maintainability of declarative process models. Still, regarding maintainability, we would like to briefly sketch the findings from a controlled experiment [22]. The results show that test cases are able to lower mental effort during model adaptations and to improve perceived quality of the resulting models. For the quality of resulting models, however, no effects could be observed. As we argue, this does not necessarily imply that test cases are not able to improve quality. Rather, a certain model complexity is required for test cases to be beneficial.

In order to investigate whether test cases are indeed beneficial above a certain model complexity, we are currently preparing a replication of the experiment described in [22]. In addition, we are preparing a case study in which TDMS will be applied in real-world modeling scenarios. Therein, we will investigate in how far the adoption of TDMS influences the communication between DE and MB.

References

1. Lenz, R., Reichert, M.: IT support for healthcare processes - premises, challenges, perspectives. DKE 61, 39–58 (2007)
2. Dumas, M., van der Aalst, W.M., ter Hofstede, A.H.: Process Aware Information Systems: Bridging People and Software Through Process Technology. Wiley-Interscience (2005)
3. Reichert, M., Dadam, P.: ADEPTflex: Supporting Dynamic Changes of Workflow without Losing Control. JIIS 10, 93–129 (1998)
4. van der Aalst, W.M.P., Weske, M.: Case handling: a new paradigm for business process support. DKE 53, 129–162 (2005)
5. Pesic, M., Schonenberg, M.H., Sidorova, N., van der Aalst, W.M.P.: Constraint-Based Workflow Models: Change Made Easy. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 77–94. Springer, Heidelberg (2007)
6. Sadiq, S.W., Orlowska, M.E., Sadiq, W.: Specification and validation of process constraints for flexible workflows. ISJ 30, 349–378 (2005)
7. Weber, B., Reichert, M., Rinderle, S.: Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. DKE 66, 438–466 (2008)
8. Pesic, M.: Constraint-Based Workflow Management Systems: Shifting Control to Users. PhD thesis, TU Eindhoven (2008)
9. 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)
10. Zugal, S., Pinggera, J., Weber, B.: Toward Enhanced Life-Cycle Support for Declarative Processes. JSME (2011), doi:10.1002/smr.554
11. van Bommel, P., Hoppenbrouwers, S.J.B.A., Proper, H.A(E.), van der Weide, T.P.: Exploring Modelling Strategies in a Meta-modelling Context. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4278, pp. 1128–1137. Springer, Heidelberg (2006)

12. Montali, M., Pesic, M., van der Aalst, W., Chesani, F., Mello, P., Storari, S.: Declarative Specification and Verification of Service Choreographies. ACM Trans. Web 4, 1–62 (2010)
13. Zugal, S., Pinggera, J., Weber, B., Mendling, J., Reijers, H.A.: Assessing the impact of hierarchy on model understandability—a cognitive perspective. In: Proc. EESSMod 2011, pp. 18–27 (2011)
14. Scaife, M., Rogers, Y.: External cognition: how do graphical representations work? Int. J. Human-Computer Studies 45, 185–213 (1996)
15. Zhang, J., Norman, D.A.: Representations in distributed cognitive tasks. Cognitive Science 18, 87–122 (1994)
16. Zhang, J.: The nature of external representations in problem solving. Cognitive Science 21, 179–217 (1997)
17. Zugal, S., Pinggera, J., Weber, B.: Assessing process models with cognitive psychology. In: Proc. EMISA 2011, pp. 177–182 (2011)
18. Reijers, H.A., Mendling, J.: A Study into the Factors that Influence the Understandability of Business Process Models. IEEE Transaction on Systems Man & Cybernetics, Part A 41, 449–462 (2011)
19. Kim, J., Lerch, F.J.: Why Is Programming (Sometimes) So Difficult? Programming as Scientific Discovery in Multiple Problem Spaces. ISR 8, 25–50 (1997)
20. Green, T.R., Petre, M.: Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. JVLC 7, 131–174 (1996)
21. Beck, K.: Test Driven Development: By Example. Addison-Wesley (2002)
22. Zugal, S., Pinggera, J., Weber, B.: The Impact of Testcases on the Maintainability of Declarative Process Models. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPMDS 2011 and EMMSAD 2011. LNBIP, vol. 81, pp. 163–177. Springer, Heidelberg (2011)
23. Pesic, M., Schonenberg, H., van der Aalst, W.: DECLARE: Full Support for Loosely-Structured Processes. In: Proc. EDOC 2007, pp. 287–298 (2007)
24. 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)
25. Pinggera, J., Zugal, S., Weidlich, M., Fahland, D., Weber, B., Mendling, J., Reijers, H.A.: Tracing the Process of Process Modeling with Modeling Phase Diagrams. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part I. LNBIP, vol. 99, pp. 370–382. Springer, Heidelberg (2012)
26. Hoppenbrouwers, S.J.B.A(S.), Lindeman, L(L.), Proper, H.A(E.): Capturing Modeling Processes – Towards the MoDial Modeling Laboratory. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4278, pp. 1242–1252. Springer, Heidelberg (2006)
27. Hoppenbrouwers, S.J., Proper, E.H., van der Weide, T.P.: Formal Modelling as a Grounded Conversation. In: Proc. LAP 2005, pp. 139–155 (2005)
28. Tort, A., Olivé, A.: An approach to testing conceptual schemas. DKE 69, 598–618 (2010)
29. Tort, A., Olivé, A.: First Steps Towards Conceptual Schema Testing. In: Proc. CAiSE Forum 2009, pp. 1–6 (2009)
30. Amyot, D., Eberlein, A.: An Evaluation of Scenario Notations and Construction Approaches for Telecommunication Systems Development. Telecommunication Systems 24, 61–94 (2003)

31. Liang, H., Dingel, J., Diskin, Z.: A comparative survey of scenario-based to state-based model synthesis approaches. In: Proc. SCESM 2006, pp. 5–12 (2006)
32. Fahland, D.: From Scenarios To Components. PhD thesis, Humboldt-Universität zu Berlin (2010)
33. 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)
34. Lamma, E., Mello, P., Montali, M., Riguzzi, F., Storari, S.: Inducing Declarative Logic-Based Models from Labeled Traces. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 344–359. Springer, Heidelberg (2007)
35. Glinz, M., Seybold, C., Meier, S.: Simulation-Driven Creation, Validation and Evolution of Behavioral Requirements Models. In: Proc. MBEES 2007, pp. 103–112 (2007)

Discovering Hierarchical Process Models Using ProM

R.P. Jagadeesh Chandra Bose[1,2], Eric H.M.W. Verbeek[1], and Wil M.P. van der Aalst[1]

[1] Department of Mathematics and Computer Science, University of Technology, Eindhoven, The Netherlands
{j.c.b.rantham.prabhakara,h.m.w.verbeek,w.m.p.v.d.aalst}@tue.nl
[2] Philips Healthcare, Veenpluis 5–6, Best, The Netherlands

Abstract. Process models can be seen as "maps" describing the operational processes of organizations. Traditional process discovery algorithms have problems dealing with fine-grained event logs and less-structured processes. The discovered models (i.e., "maps") are spaghetti-like and are difficult to comprehend or even misleading. One of the reasons for this can be attributed to the fact that the discovered models are flat (without any hierarchy). In this paper, we demonstrate the discovery of hierarchical process models using a set of interrelated plugins implemented in ProM.[1] The hierarchy is enabled through the automated discovery of abstractions (of activities) with domain significance.

Keywords: process discovery, process maps, hierarchical models, abstractions, common execution patterns.

1 Introduction

Process discovery is one of the three main types of process mining [1]. A discovery technique takes an event log and produces a model without using any *apriori* information e.g., the α-algorithm discovers a Petri net based on sequences of events [2]. We have applied process mining techniques in over 100 organizations. These practical experiences revealed two problems: (a) processes tend to be less structured than what stakeholders expect, and (b) events logs contain fine-grained events whereas stakeholders would like to view processes at a more coarse-grained level. As a result, the discovered models are often incomprehensible (spaghetti-like) and add little value. This can be attributed to the fact that the majority of techniques pertains to the discovery of control-flow models that are "flat" [2,3,4,5]. A notable exception is the Fuzzy miner [6]. Flat models have inherent limitations and are one of the primary sources of incomprehensibility. For a log with $|\mathcal{A}|$ event classes (activities), a flat model can be viewed as a graph containing $|\mathcal{A}|$ nodes with edges corresponding to the dependency between activities defined by the execution behavior in the log. Graphs become quickly

[1] ProM is an extensible framework that provides a comprehensive set of tools/plugins for the discovery and analysis of process models from event logs. See http://www.processmining.org for more information and to download ProM.

overwhelming and unsuitable for human perception and cognitive systems even if there are a few dozens of nodes [7].

In [8], we showed that common execution patterns (e.g., tandem arrays, maximal repeats etc.) manifested in an event log can be used to create powerful *abstractions* (the abstractions uncovered have strong domain significance from a functionality point of view). These abstractions are used in our *two-phase approach to process discovery* [9]. The first phase comprises of pre-processing the event log based on abstractions (bringing the log to the desired level of granularity) and the second phase deals with discovering the process maps while providing a seamless zoom-in/out facility. The two-phase approach to process discovery has been implemented as a set of interrelated plugins in the ProM framework. In this paper, we demonstrate the discovery of hierarchical process models using a chain of these plugins.

Running Example. We use the workflow of a simple digital photo copier as our running example. The copier supports photocopying, scanning and printing of documents in both color and gray modes. The scanned documents can be sent to the user via email or FTP. Upon receipt of a job, the copier first generates an image of the document and subsequently processes the image to enhance its quality. Depending on whether the job request is for a copy/scan or print job, separate procedures are followed to generate an image. For print requests, the document is first interpreted and then a rasterization procedure is followed to form an image. The image is then written on the drum, developed, and fused on to the paper. Fig. 15 in Appendix A depicts the high-level workflow of the digital photo copier represented as an YAWL [10] model. This high-level workflow contains the composite tasks (sub-processes) *capture image*, *rasterize image*, *image processing* and *print image*. Fig. 16 in Appendix A depicts the workflow of the image processing sub-process. This sub-process contains another composite task, viz., *half toning*, which is depicted in Fig. 17 in Appendix A. Fig. 18 in Appendix A depicts the *fusing* sub-process within *print image*.

We have modeled this workflow of the copier in CPN tools [11] and generated event logs by simulation [12]. CPN tools lets the users model processes and have more control about the properties of the event logs. This is achieved by varying the parameters of the model and the parameters for simulation. The advantage of using synthetic event logs is that we can conduct various controlled experiments. We also applied the approach presented in this paper to several real-life event logs. These experiments confirm our findings based on the simulated model used in this paper. To illustrate our approach we use one event log generated for the copier. The event log consists of 100 process instances, 76 event classes and $40,995$ events[2]. The event log contains fine-grained events pertaining to different procedures (e.g., image processing, image generation etc.) mentioned above. An analyst may not be interested in such low level details. We demonstrate the discovery of the workflow at various levels of abstractions for this event log.

[2] The event log is available at:
`http://www.win.tue.nl/~jcbose/DigitalCopier.xes.gz`.

The remainder of this paper is organized as follows. Section 2 presents an overview of the two phase approach to process discovery. Section 3 describes how the two-phase approach can be used to discover hierarchical processes. Section 4 explains the pattern abstractions plugin in ProM that assists in the abstraction of events and the transformation of an event log to a desired level of granularity. Section 5 explains the enhanced fuzzy miner plugin in ProM that enables the discovery of process maps. We present the experimental results in Section 6. Section 7 concludes the paper.

2 Two-Phase Approach to Process Discovery

The first phase in the two-phase approach to process discovery involves the simplification of the log to a desired level of granularity. The second phase involves the discovery of models from this simplified log.

Phase-1: Preprocessing Log

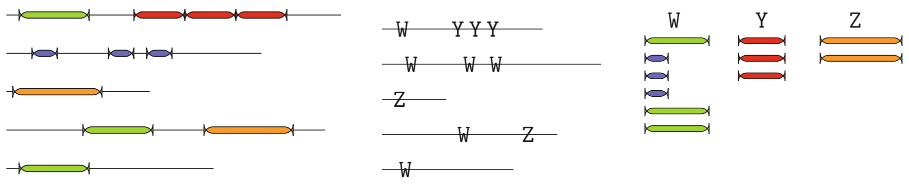
In this phase, the log is simplified based on the desired traits of the context of analysis. Some notion of abstraction of the low-level events to a desired level of granularity needs to be defined. In some applications and contexts, this can be provided by analysts based on domain knowledge e.g., there can be certain activities that demarcate the execution of a particular medical procedure on an X-ray machine. A sequence of events between these activities pertains to an execution sequence of this procedure e.g., the sequence of activities between `Start Fluoroscopy` and `Stop Fluoroscopy` define an instance of a `Fluoroscopy` procedure applied to a patient on an X-ray machine. Alternatively, abstractions can be defined by uncovering common execution patterns in the log and establishing relationships between them as discussed in [8]. These common execution patterns typically capture a sub-process/functionality. Such subprocess behavior in its totality can be captured as an *abstract activity*. Abstractions can be considered as a mapping, $\mathcal{M} \subseteq 2^{\Sigma} \times \mathcal{A}$, between the *original alphabet*, Σ, of the event log, and an *abstract alphabet* $\mathcal{A}$. An example mapping is $\mathcal{M} = \{(\{\mathtt{a},\mathtt{b}\},\mathtt{x}),(\{\mathtt{b},\mathtt{c},\mathtt{d}\},\mathtt{y}),(\{\mathtt{e}\},\mathtt{z}),(\{\mathtt{d}\},\mathtt{z})\}$. This mapping is analogous to the grouping and tagging of streets as a town/city in cartography and to the selection of a desired perspective of viewing maps (restaurant maps vs. fuel station maps)[3]. Each $(B,\mathtt{a}) \in \mathcal{M}$ may reflect a set of sequences $\mathbf{s} \in B^{+}$ capturing the set of patterns defined by the alphabet B for the abstraction $\mathtt{a}$.

Using this mapping, the original event log $\mathcal{L}$, is transformed into an *abstract log* $\mathcal{L}'$. Each trace $\mathbf{t} \in \mathcal{L}$ is transformed into a corresponding trace $\mathbf{t}' \in \mathcal{L}'$. At the same time, we create one sub-log for each abstract activity $\mathtt{a} \in \mathcal{A}$. The basic

[3] One can also take the analogy of *atoms* and *molecules* from chemistry. The individual activities in the original event log are atoms while the mapping associates groups of atoms (activities) to molecules (abstract activities).

idea of the transformation algorithm is to scan each trace from left to right and in the process determine whether there exists a pattern in the trace for which an abstraction is defined. If such a pattern exists, the manifestation of the pattern in the trace is replaced by its abstract activity and simultaneously the manifestation of the pattern that is replaced is added as a trace (process instance) in the sub-log corresponding to that abstract activity. It could be the case that the manifestation of patterns is disturbed by the presence of concurrent activities and/or noise in a trace. Therefore, we consider not just exact but also inexact (non-continuous and approximate) manifestations of patterns to deal with such scenarios. Fig. 1 depicts the general idea of the event log transformation.

$\mathcal{D} = \bigcup_{(B,\mathsf{a})\,\in\mathcal{M}} B$ denotes the set of activities in Σ for which a mapping is defined. The activities in $\Sigma \setminus \mathcal{D}$ being not involved in the definition of mapping indicate activities that are insignificant from the context of analysis and are filtered from $\mathbf{t}$ during this transformation. The transformation of logs can be associated to the concept of *artifacts* [13] or *proclets* [14] in business processes. Artifact-centric process models partition a monolithic process model into smaller loosely-coupled process fragments, each describing the life-cycle of a concrete and identifiable artifact.



(a) original log with common execution patterns (b) transformed log and the sub-logs for each abstraction

Fig. 1. Transformation of the original log into an abstracted log. Also, one sub-log is created for each abstract activity. `W`, `Y`, and `Z` are abstract activities.

Phase-2: Mining Maps

The second phase is to mine a process model on the abstracted (transformed) log. The mapping defined in Phase-1 induces a hierarchy over the abstract activities. Upon zooming into an abstract activity, a process model depicting the subprocess captured by this abstract activity can be shown. The sub-log for the abstract activity is used to create this sub-process model. Multiple levels of hierarchy can be obtained by a repetitive application of Phase-1 i.e., abstractions are defined over the transformed log (pre-processed log with abstractions) obtained in iteration i in iteration $i+1$. This results in new abstractions to be defined over existing abstractions, thus inducing a hierarchy.

Fig. 2 illustrates the difference between the traditional approach to process discovery and our two-phase approach. Note that the process model (map) discovered using the two-phase approach is much simpler.

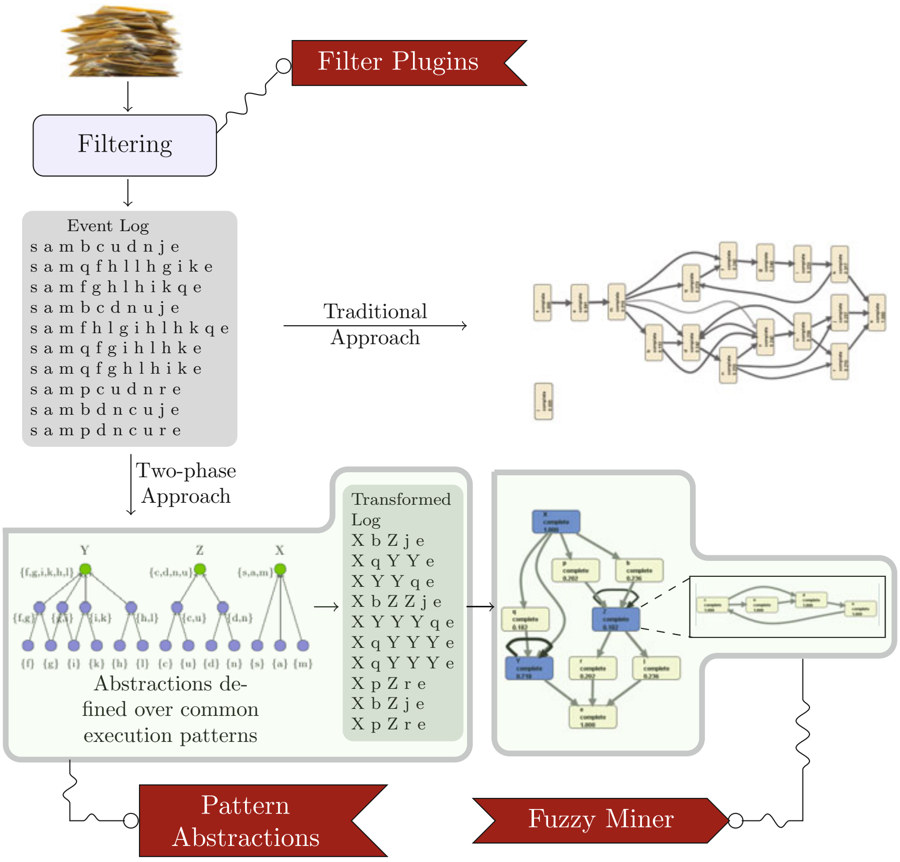


Fig. 2. Traditional approach versus our two-phase approach to process discovery. ProM plugins are used to filter the event log. ProM's *Pattern Abstractions* plugin and the *Fuzzy Miner* plugin are used to realize simple and intuitive models.

3 Discovering Hierarchical Processes

The *two-phase approach to process discovery* described in Section 2 enables the discovery of hierarchical process models [9]. In this paper, we demonstrate this using a chain of plugins implemented in ProM. The chain of plugins and their order of application is illustrated in Fig. 3. The event log may first be cleansed using some simple filters (e.g., adding artificial start/end events, filtering events of a particular transaction type such as considering only 'complete' events, etc.).

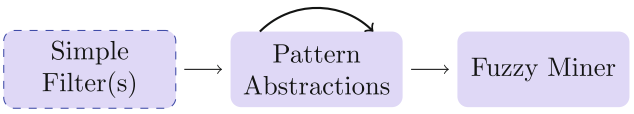


Fig. 3. The chaining of plugins that enables the discovery of hierarchical process models

The *Pattern Abstractions* plugin is then applied on this filtered log one or several times. The Pattern Abstractions plugin has been implemented as a *log visualizer* in ProM and caters to *the discovery of common execution patterns, the definition of abstractions over them, and the pre-processing of the event log with these abstractions*. The transformed log (pre-processed log with abstractions) obtained in iteration i is used as the input for the Pattern Abstractions plugin in iteration $i + 1$. It is this repetitive application of the Pattern Abstractions plugin that enables the discovery of multiple levels of hierarchy (new abstractions can be defined over existing abstractions). During the pre-processing phase, for each defined abstraction, the Pattern Abstractions plugin generates a sub-log that captures the manifestation of execution patterns defined by that abstraction as its process instances. The Fuzzy Miner plugin [6] is then applied on the transformed log obtained after the last iteration. The Fuzzy Miner plugin in ProM has been enhanced to utilize the availability of sub-logs for the defined abstractions. Process models are discovered for each of the sub-logs and are displayed upon zooming in on its corresponding abstract activity.

In the next two sections, we explain the functionality and features of the Pattern Abstractions and the (Enhanced) Fuzzy Miner plugins.

4 Pattern Abstractions Plugin

The basic building blocks of the Pattern Abstractions plugin are shown in Fig. 4. Figures 5 and 6 illustrate these building blocks.

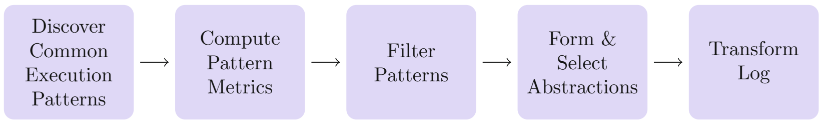
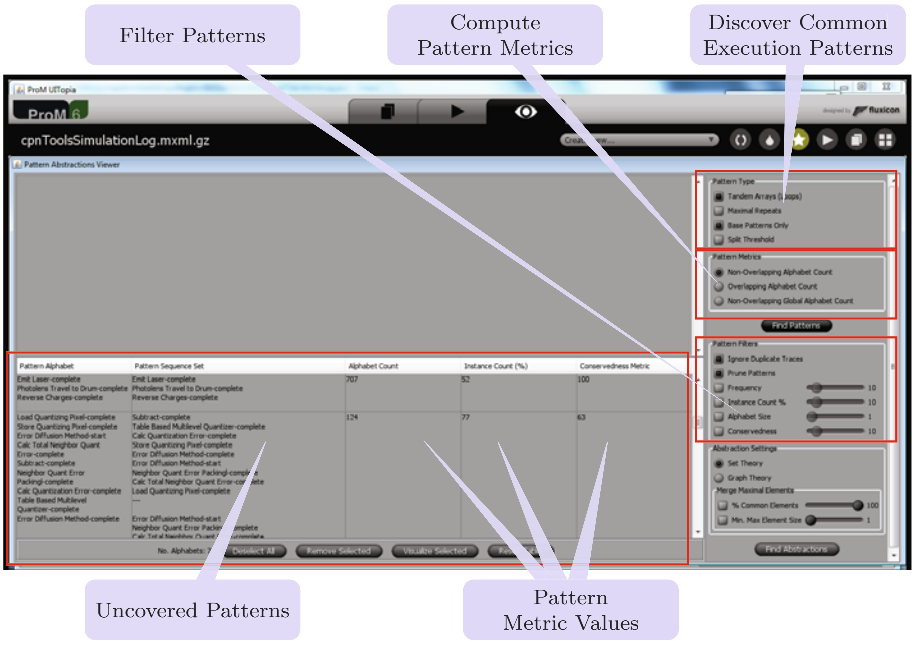

Discover Common Execution Patterns
Compute Pattern Metrics
Filter Patterns
Form & Select Abstractions
Transform Log

Fig. 4. Building blocks of the Pattern Abstractions plugin

– *Discover Common Execution Patterns:* The Pattern Abstractions plugin supports the discovery of tandem arrays (loop patterns) and maximal repeats (common subsequence of activities within a process instance or across process instances) [8]. These can be uncovered in *linear* time and space with respect to the length of the traces.



Pattern Alphabet	Pattern Sequence Set	Alphabet Count	Instance Count (%)	Conservedness Metric
Emit Laser-complete Photolens Travel to Drum-complete Reverse Charges-complete	Emit Laser-complete Photolens Travel to Drum-complete Reverse Charges-complete	707	52	100
Load Quantizing Pixel-complete Store Quantizing Pixel-complete Error Diffusion Method-start Calc Total Neighbor Quant Error-complete Subtract-complete Neighbor Quant Error Packingl-complete Calc Quantization Error-complete Table Based Multilevel Quantizer-complete Error Diffusion Method-complete	Subtract-complete Table Based Multilevel Quantizer-complete Calc Quantization Error-complete Store Quantizing Pixel-complete Error Diffusion Method-complete Error Diffusion Method-start Neighbor Quant Error Packingl-complete Calc Total Neighbor Quant Error-complete Load Quantizing Pixel-complete — Error Diffusion Method-start Neighbor Quant Error Packingl-complete	124	77	63

No. Alphabets: 7

Pattern Type
- [✓ Tandem Arrays (Loops)]
- Maximal Repeats
- [✓ Base Patterns Only]
- Split Threshold

Pattern Metrics
- [✓ Non-Overlapping Alphabet Count]
- Overlapping Alphabet Count
- Non-Overlapping Global Alphabet Count

Pattern Filters
- [✓ Ignore Duplicate Traces]
- [✓ Prune Patterns]
- Frequency 10
- Instance Count % 10
- Alphabet Size 1
- Conservedness 10

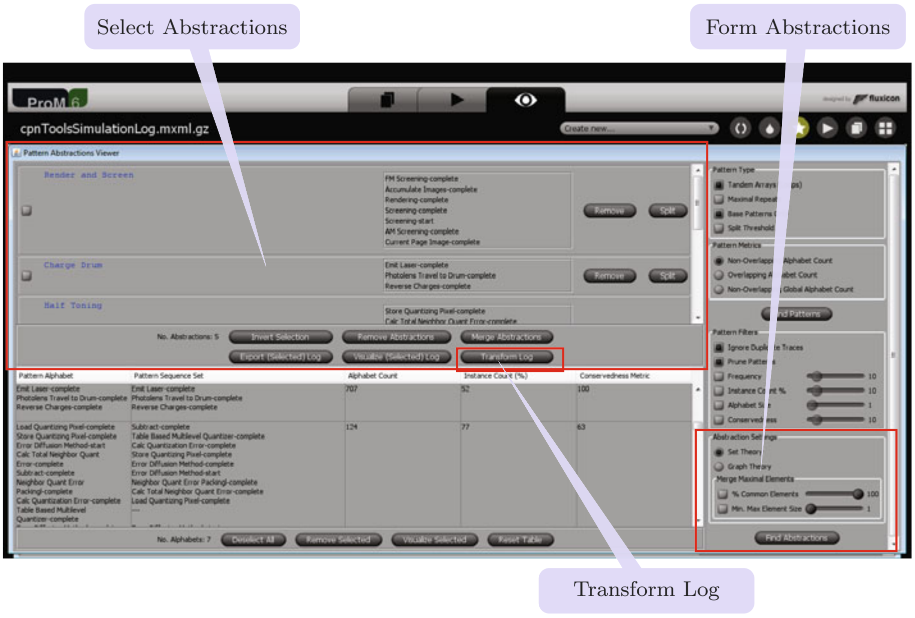
Abstraction Settings
- [✓ Set Theory]
- Graph Theory

Merge Maximal Elements
- % Common Elements 100
- Min. Max Element Size 1

Fig. 5. The discovery of common execution patterns, computation of pattern metrics, filtering and inspection of patterns in the Pattern Abstractions plugin

- *Compute Pattern Metrics:* Various metrics (e.g, overlapping and non-overlapping frequency counts, instance count, etc.) to assess the significance of the uncovered patterns are supported.
- *Filter Patterns:* It could be the case that too many patterns are uncovered from the event log. To manage this, features to filter patterns that are less significant are supported.
- *Form and Select Abstractions:* Abstractions are defined over the filtered patterns. Patterns that are closely related are grouped together to form abstractions. The approach for forming abstractions is presented in [8]. Furthermore, various features to edit/select abstractions such as merging two or more abstractions and deleting activities related to a particular abstraction are supported. Fig. 6 depicts a few abstractions defined over loop patterns for the copier event log e.g., *half-toning*, a procedure for enhancing the image quality, is uncovered as an abstraction.
- *Transform Log:* The event log is pre-processed by replacing activity subsequences corresponding to abstractions. A replaced activity subsequence is captured as a process instance in the sub-log for the corresponding abstract activity.

At any iteration, if n abstractions are selected, the Pattern Abstractions plugin generates a transformed log, and n sub-logs (one for each of the n chosen abstractions). We recommend to process for loop patterns in the initial iterations and maximal repeats in the subsequent iterations. For the example event log, we have performed three iterations. The transformed log after the third iteration has 19 event classes and 1601 events. In the process, we have defined various abstractions such as *half-toning, image processing, capture image*, etc.



Render and Screen
FM Screening-complete
Accumulate Images-complete
Rendering-complete
Screening-complete
Screening-start
AM Screening-complete
Current Page Image-complete

Charge Drum
Emit Laser-complete
Photolens Travel to Drum-complete
Reverse Charges-complete

Half Toning
Store Quantizing Pixel-complete
Calc Total Neighbor Quant Error-complete

No. Abstractions: 5

Pattern Alphabet	Pattern Sequence Set	Alphabet Count	Instance Count (%)	Conservedness Metric
Emit Laser-complete Photolens Travel to Drum-complete Reverse Charges-complete	Emit Laser-complete Photolens Travel to Drum-complete Reverse Charges-complete	707	52	100
Load Quantizing Pixel-complete Store Quantizing Pixel-complete Error Diffusion Method-start Calc Total Neighbor Quant Error-complete Subtract-complete Neighbor Quant Error Packingl-complete Calc Quantization Error-complete Table Based Multilevel Quantizer-complete	Subtract-complete Table Based Multilevel Quantizer-complete Calc Quantization Error-complete Store Quantizing Pixel-complete Error Diffusion Method-complete Error Diffusion Method-start Neighbor Quant Error Packingl-complete Calc Total Neighbor Quant Error-complete Load Quantizing Pixel-complete ----	124	77	63

No. Alphabets: 7

Pattern Type
- [✓] Tandem Arrays (
- [ ] Maximal Repeat
- [✓] Base Patterns
- [ ] Split Threshold

Pattern Metrics
- (●) Non-Overlapp Alphabet Count
- ( ) Overlapping A abet Count
- ( ) Non-Overlapp Global Alphabet Count

Pattern Filters
- [✓] Ignore Dupli te Traces
- [✓] Prune Patte s
- [ ] Frequency 10
- [ ] Instance Co nt % 10
- [ ] Alphabet Si e 1
- [ ] Conservedn ess 10

Abstraction Settings
- (●) Set Theory
- ( ) Graph Theory

Merge Maximal Elements
- [ ] % Common Elements 100
- [ ] Min. Max Element Size 1

Fig. 6. The generation and selection of abstractions in the Pattern Abstractions plugin

The Pattern Abstractions plugin supports additional features such as visualizing patterns and exporting the traces that contain the patterns.

5 (Enhanced) Fuzzy Miner Plugin

The Fuzzy Miner [6,15] is a process discovery technique that mines an event log for a family of process models using a "map" metaphor. As many maps exist that show the city of Amsterdam at different levels of abstraction, also different maps exist for a process model mined from an event log. In this map metaphor, an object of interest in Amsterdam (like the Rijksmuseum or the Anne Frank House) corresponds to a node in the process model, where streets (like the Kalverstraat or the PC Hooftstraat) correspond to edges in the model. For sake of convenience, we call a single map a *fuzzy instance* whereas we call a family of maps (like all Amsterdam maps) a *fuzzy model*.

Like high-level maps only show major objects of interest and major streets, high-level fuzzy instances show only major elements (nodes and edges). For this purpose, the Fuzzy Miner computes from the log a significance weight for every element and an additional correlation weight for every edge. The higher these weights are, the more major the element is considered to be. Furthermore, the Fuzzy Miner uses a number of thresholds: only elements that meet these thresholds are shown. As such, these thresholds correspond to the required level of abstraction: the higher these thresholds are, the higher the level of abstraction is. For sake of completeness we mention here that a fuzzy instance may contain clusters of minor nodes: If some objects of interest on the Amsterdam map are too minor to be shown by themselves on some map, they may be shown as a single (and major) object provided that they are close enough. For this reason, the Fuzzy Miner first attempts to cluster minor nodes into major cluster nodes, and only if that does not work it will remove the minor node from the map.

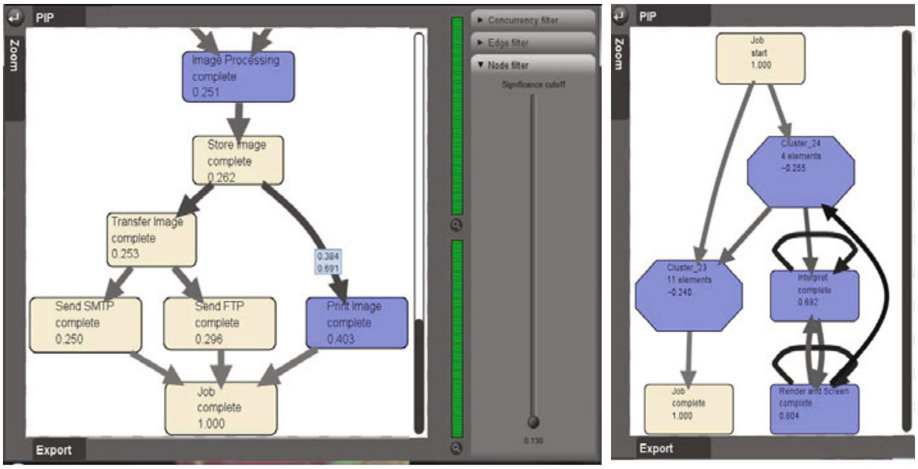


Fig. 7. Fuzzy model and instance

Fig. 7 shows an example fuzzy model (left-hand side) and fuzzy instance (right-hand side). Note that both views show a fuzzy instance, but the fuzzy model view allows the user to change the thresholds (by changing the sliders) whereas the fuzzy instance view does not. The significance of a node is displayed as part of its label (for example, the node "Transfer Image" has a significance of 0.253), the significance of an edge is visualized using its wideness (the wider the edge, the more significant it is), and the correlation of an edge is visualized using its color contrast (the darker the edge is, the more correlated its input node and its output node are). The octagonal shaped nodes in the right-hand side view correspond to the cluster nodes (one of the cluster nodes contain 4 activities and the other contains 11 activities). All activities on the left hand side except "Job Complete" are contained in a cluster node on the right. Apparently, the significance weights for these nodes (0.262, 0.253, 0.250, 0.296 and 0.403) were too low to be shown, which indicates that the corresponding threshold was set to at least 0.403. Furthermore, the node “Interpret” (on the right) is highly self-correlated, whereas the nodes “Transfer Image” and “Send SMTP” (on the left) are moderately correlated.

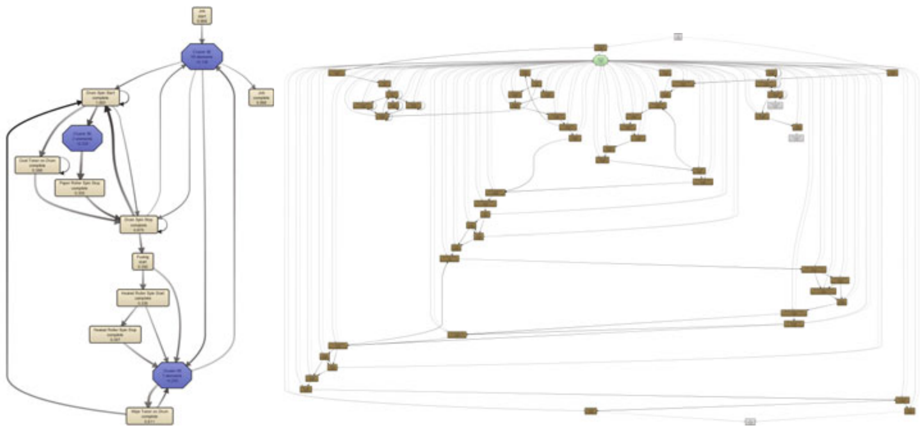
The Fuzzy Miner has been enhanced (called as the Fuzzy Map miner) to utilize the availability of sub-logs obtained from the Pattern Abstractions plugin for the chosen abstractions. Fuzzy models are discovered for each of the sub-logs and are displayed upon zooming in on its corresponding abstract activity. Abstract activities are differentiated from other activities by means of a distinct color (a darker shade of blue, see also Fig. 7). Thus, the enhanced fuzzy miner gives the user more control over the discovery process than the classical fuzzy miner [6].

6 Experimental Results and Discussion

In this section, we demonstrate the discovery of hierarchical processes using the concepts presented in this paper firstly on the synthetic log of the digital copier example and later on a real-life case study. We contrast the process models discovered using the proposed approach with the models uncovered using the classical Fuzzy miner. Fig. 8(a) depicts the process model mined on the digital copier event log using the classical Fuzzy miner [6]. As discussed in Section 5, the Fuzzy miner plugin groups activities into cluster nodes (blue colored nodes in Fig. 8(a)) based on their significance. Fig. 8(b) depicts the process model obtained upon zooming one of the cluster nodes containing 55 activities. As we can see, the model is Spaghetti-like and hard to comprehend. The cluster nodes formed by the classical fuzzy miner do not have any semantic significance with respect to the domain/application. In other words, the classical fuzzy miner poses the risk of aggregating unrelated nodes together in a cluster (using the map metaphor, this is similar to streets in Eindhoven being clustered along with streets in Amsterdam).

Fig. 9 depicts the top-level process model of the copier example discovered using the chain of plugins proposed in this paper. This model is generated from the transformed log obtained after the third iteration of Pattern Abstractions plugin. The upper branch of the process model corresponds to the creation of the document image for print requests while the lower branch corresponds to image creation for copy/scan requests. The two branches meet after the image is formed and the image is subjected to some image processing functionality. The document is then printed or sent to the user via email or FTP. The lower level details of image creation, image processing, print image have been abstracted in this model. *The Pattern Abstractions plugin enables the discovery of such abstractions with strong domain (functional) significance.* Note the similarity of this discovered model with the original model in Fig. 15. The *Interpret* and *Render and Screen* abstractions in Fig. 9 are within the Rasterize Image composite task of Fig. 15.

Upon zooming in on the *Image Processing* abstraction, the process model depicted in Fig. 10 is shown. Note the similarity of this discovered model with that of the original model in Fig. 16. This sub-process in turn contains another



(a) top-level model (b) process model upon zooming the cluster node 'cluster 90'

Fig. 8. Process model mined using the classical fuzzy miner

abstract activity viz., *Half Toning* (the level of hierarchy is two). Zooming in on this abstract activity displays the sub-process defining it as depicted in Fig. 10. Note the similarity of this discovered model for half toning with that of the original model in Fig. 17. Fig. 11 depicts two other abstractions viz., *Interpret* and *Fusing*. Note the similarity of the discovered model for *Fusing* with that of the original model in Fig. 18.

In this fashion, using the chain of plugins presented in this paper, one can discover hierarchical process models. We next present the results of applying this approach in a real-life case study. The case study was performed in close collaboration with *Philips Healthcare* and pertains to the analysis of event logs generated by the *X-ray machines*. More specifically, we considered an event log capturing

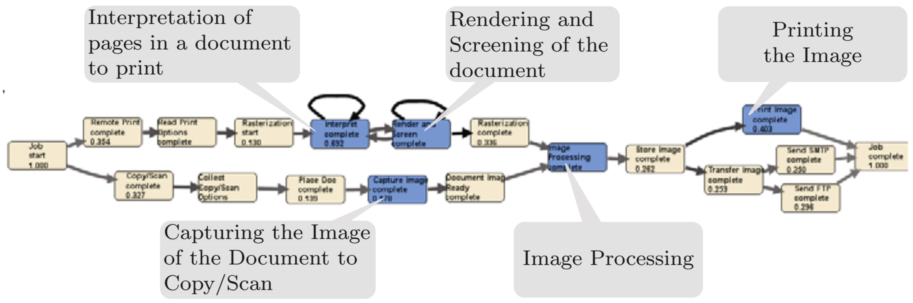

Interpretation of pages in a document to print
Rendering and Screening of the document
Printing the Image
Job start 1.000
Remote Print complete 0.354
Read Print Options complete
Rasterization start 0.130
Interpret complete 0.692
Render and Screen complete
Rasterization complete 0.335
Image Processing complete
Store Image complete 0.262
Print Image complete 0.403
Transfer Image complete 0.253
Send SMTP complete 0.250
Send FTP complete 0.298
Job complete 1.000
Copy/Scan complete 0.327
Collect Copy/Scan Options
Place Doc complete 0.139
Capture Image complete 0.178
Document Image Ready complete
Capturing the Image of the Document to Copy/Scan
Image Processing

Fig. 9. The top level process model of the copier event log. Blue (dark colored) nodes are abstract activities that can be zoomed in. Upon zooming in, the sub-process defining the abstraction is shown.

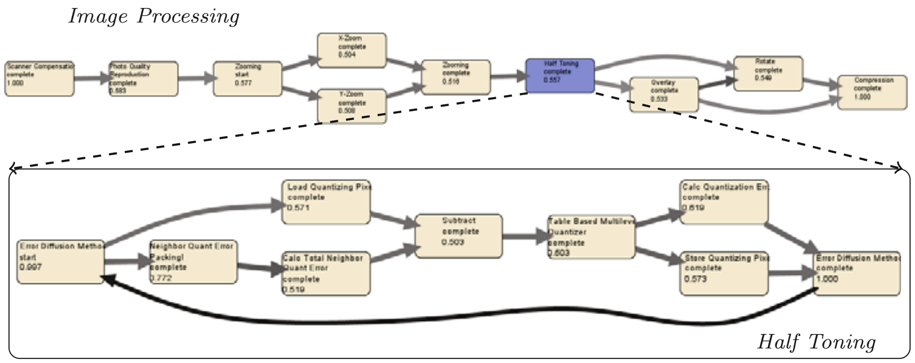


Image Processing
Scanner Compensatio complete 1.000
Photo Quality Reproduction complete 0.583
Zooming start 0.577
X-Zoom complete 0.504
Y-Zoom complete 0.508
Zooming complete 0.516
Half Toning complete 0.557
Overlay complete 0.533
Rotate complete 0.549
Compression complete 1.000
Error Diffusion Metho start 0.997
Neighbor Quant Error Packingl complete 0.772
Load Quantizing Pix complete 0.571
Calc Total Neighbor Quant Error complete 0.519
Subtract complete 0.503
Table Based Multileve Quantizer complete 0.603
Calc Quantization Err complete 0.619
Store Quantizing Pix complete 0.573
Error Diffusion Metho complete 1.000
Half Toning

Fig. 10. The sub-process captured for the abstraction 'Image Processing' (in the top-level model). This sub-process in turn contains another abstraction viz., 'Half Toning'. Upon zooming in on 'Half Toning', the sub-process defining that is shown.

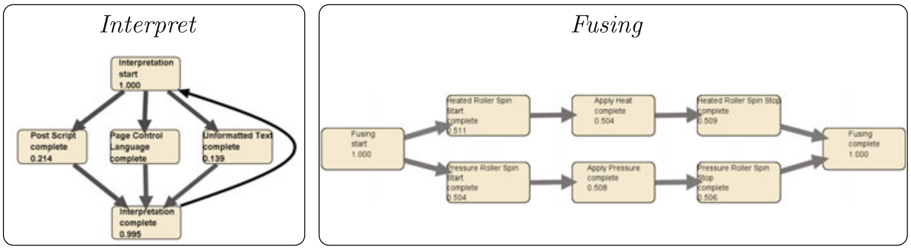

Interpret
Interpretation start 1.000
Post Script complete 0.214
Page Control Language complete
Unformatted Text complete 0.139
Interpretation complete 0.995
Fusing
Fusing start 1.000
Heated Roller Spin Start complete 0.511
Apply Heat complete 0.504
Heated Roller Spin Stop complete 0.509
Pressure Roller Spin Start complete 0.504
Apply Pressure complete 0.508
Pressure Roller Spin Stop complete 0.506
Fusing complete 1.000

Fig. 11. The sub-processes for the abstractions 'Interpret' and 'Fusing'. 'Interpret' is an abstract activity at the top-level of the process model while 'Fusing' is an abstract activity underneath the 'Print Image' abstraction.

the activities performed by field service engineers during one of the part replacements on X-ray machines. The event log contains 113 cases and 76, 754 events referring to 92 activities. Fig. 12 depicts the process model uncovered on this log using the classical Fuzzy miner. Again, we see an incomprehensible spaghetti-like model. We uncover common execution patterns, define abstractions, and use our two-phase approach to alleviate this problem. The transformed log (with abstractions) contains 113 cases and 10, 387 events referring to 20 activities. Fig. 13 depicts the top-level process model discovered using the Fuzzy map miner on this transformed log. We can see that the model discovered using our two-phase approach is simpler and more comprehensible. Fig. 14(a) depicts the sub-process for the abstract activity Fluoroscopy, Exposure, and Viewing while Fig. 14(b) depicts the sub-process for the abstract activity Beam Limitation.

We have used this approach on several other real-life event logs as well (e.g., see [16]). Our experiences show that the automated abstractions uncovered by the Pattern Abstractions plugin have strong domain (functional) significance. This can be attributed to that fact that these abstractions are obtained by exploiting the common execution patterns in the event log. Common subsequences

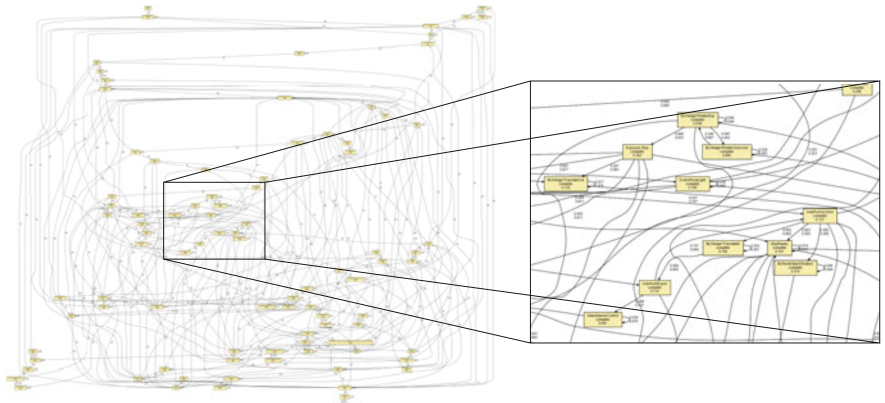


Fig. 12. Process model discovered using the classical Fuzzy miner on the event log capturing the activities of field service engineers during one of the part replacements in X-ray machines

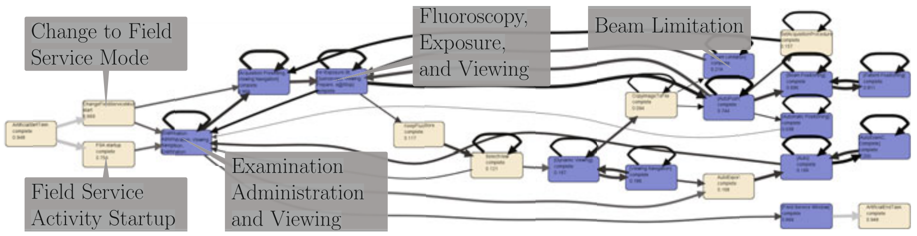
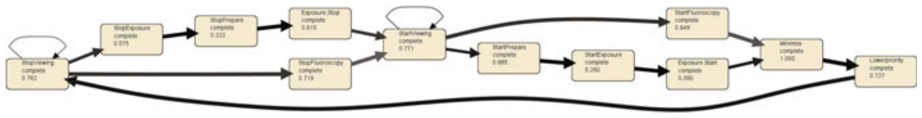
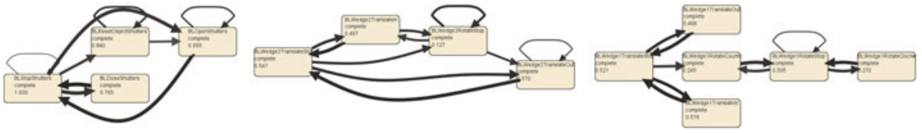

Change to Field Service Mode
Fluoroscopy, Exposure, and Viewing
Beam Limitation
Field Service Activity Startup
Examination Administration and Viewing

Fig. 13. The top-level of the hierarchical process model using the approach proposed in this paper on the event log capturing the activities of field service engineers during one of the part replacements in X-ray machines



(a) Fluoroscopy, Exposure and Viewing



(b) Beam Limitation

Fig. 14. The sub-processes corresponding to the abstract activities Fluoroscopy, Exposure and Viewing and Beam Limitation

of activities in an event log that are found to recur within a process instance or across process instances tend to have some domain (functional) significance. One of the limitations of our approach is the semi-automated means of defining abstractions. Currently, the plugin requires the manual merging/pruning of patterns. In the future, we would like to automate some of these functionalities.

7 Conclusions

We demonstrated the discovery of hierarchical process models using a chain of plugins implemented in ProM. The repetitive application of Pattern Abstractions plugin enables the discovery of multiple levels of hierarchy. We can use this approach to create comprehensible process maps.

Acknowledgments. R.P.J.C. Bose and W.M.P. van der Aalst are grateful to Philips Healthcare for funding the research in process mining.

References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer-Verlag New York Inc. (2011)
2. van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128–1142 (2004)
3. Weijters, A.J.M.M., van der Aalst, W.M.P.: Rediscovering Workflow Models From Event-based Data Using Little Thumb. Integrated Computer-Aided Engineering 10(2), 151–162 (2003)
4. van der Werf, J.M.E.M., van Dongen, B.F., Hurkens, C.A.J., Serebrenik, A.: Process Discovery Using Integer Linear Programming. In: van Hee, K.M., Valk, R. (eds.) PETRI NETS 2008. LNCS, vol. 5062, pp. 368–387. Springer, Heidelberg (2008)
5. van Dongen, B.F., Alves de Medeiros, A.K., Wen, L.: Process Mining: Overview and Outlook of Petri Net Discovery Algorithms. In: Jensen, K., van der Aalst, W.M.P. (eds.) ToPNoC II. LNCS, vol. 5460, pp. 225–242. Springer, Heidelberg (2009)
6. 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)
7. Görg, C., Pohl, M., Qeli, E., Xu, K.: Visual Representations. In: Kerren, A., Ebert, A., Meyer, J. (eds.) Human-Centered Visualization Environments 2006. LNCS, vol. 4417, pp. 163–230. Springer, Heidelberg (2007)
8. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Abstractions in Process Mining: A Taxonomy of Patterns. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 159–175. Springer, Heidelberg (2009)
9. Li, J., Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Mining Context-Dependent and Interactive Business Process Maps Using Execution Patterns. In: zur Muehlen, M., Su, J. (eds.) BPM 2010 Workshops. LNIBIP, vol. 66, pp. 109–121. Springer, Heidelberg (2011)
10. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: Yet Another Workflow Language. Information Systems 30(4), 245–275 (2005)

11. Jensen, K., Kristensen, L.M.: Colored Petri Nets: Modeling and Validation of Concurrent Systems. Springer-Verlag New York Inc. (2009)
12. Medeiros, A.K.A.D., Günther, C.W.: Process Mining: Using CPN Tools to Create Test Logs for Mining Algorithms. In: Proceedings of the Sixth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, pp. 177–190 (2005)
13. Nigam, A., Caswell, N.S.: Business Artifacts: An Approach to Operational Specification. IBM Systems Journal 42(3), 428–445 (2003)
14. van der Aalst, W.M.P., Barthelmess, P., Ellis, C.A., Wainer, J.: Proclets: A Framework for Lightweight Interacting Workflow Processes. International Journal of Cooperative Information Systems 10(4), 443–482 (2001)
15. Xia, J.: Automatic Determination of Graph Simplification Parameter Values for Fuzzy Miner. Master's thesis, Eindhoven University of Technology (2010)
16. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Analysis of Patient Treatment Procedures: The BPI Challenge Case Study. Technical Report BPM-11-18, BPMCenter.org (2011), `http://bpmcenter.org/wp-content/uploads/reports/2011/BPM-11-18.pdf`

A YAWL Models of the Digital Photo Copier

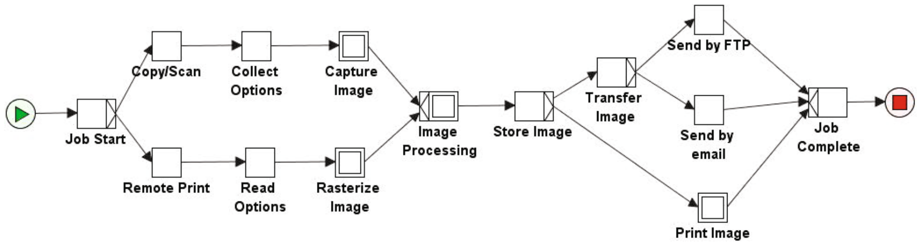


Fig. 15. High-level model of the digital photo copier. The digital copier supports two functionalities viz., copy/scan and print. Documents are interpreted and converted into an image before they are printed. Scanned images of the copy/scan jobs are sent to the user via email or ftp.

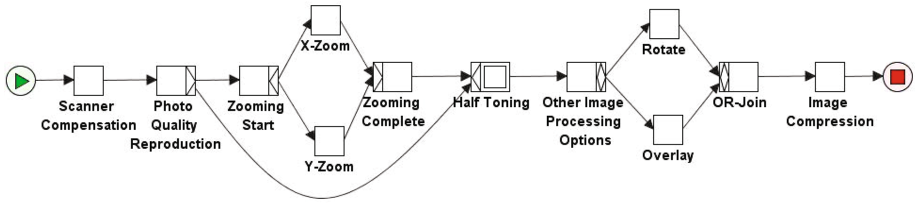


Fig. 16. The image processing sub-process. This subprocess supports operations such as zooming, rotating and overlay and attempts at improving the quality of images.

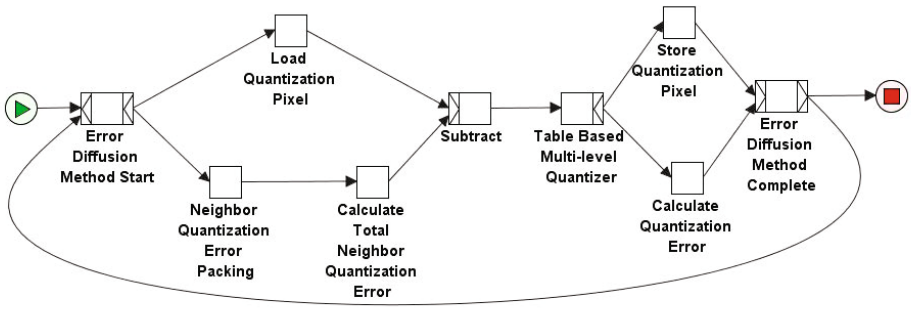

Load Quantization Pixel
Error Diffusion Method Start
Neighbor Quantization Error Packing
Calculate Total Neighbor Quantization Error
Subtract
Table Based Multi-level Quantizer
Store Quantization Pixel
Calculate Quantization Error
Error Diffusion Method Complete

Fig. 17. The half-toning sub-process for image representation. Half toning is a technique that simulates continuous tone imagery through the use of equally spaced dots of varying size.

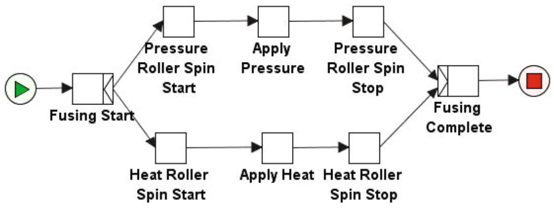


Fig. 18. The fusing sub-process within print image

Diagen: A Model-Driven Framework for Integrating Bioinformatic Tools

Maria José Villanueva, Francisco Valverde, Ana M. Levín, and Oscar Pastor Lopez

Software Production Methods Research Center
Universitat Politècnica de València
Camino de Vera S/N 46022, Valencia, Spain
{mvillanueva,fvalverde,alevin,opastor}@pros.upv.es

Abstract. Nowadays, the diagnosis of disease based on genomic information is feasible by searching genetic variations on DNA sequences. However, geneticists struggle with bioinformatic tools that are supposed to simplify DNA sequence analysis. As a universal tool to support every requirement is far from be implemented, geneticists themselves must solve the data exchange among several tools. Due to the fact that there are no standards to support this integration task, it must be managed in every analysis. This paper addresses this integration by means of a model-driven framework. The Diagen framework is a software implementation based on conceptual modeling principles that formalizes data exchange and simplifies bioinformatic tool integration. First, we analyze how conceptual modeling can be used to deal with data exchange among tools. Then, the presented framework is used to search for variations on the BRCA2 gene using real DNA samples and a set of specific bioinformatic tools.

Keywords: Model-Driven Development, Tool Integration, DNA sequence analysis.

1 Introduction

Recent genetic discoveries have opened the door to personalized disease diagnosis based on DNA sequence analysis. A DNA sample, extracted from i.e. blood, is treated by sequencing machines and, then, a DNA sequence is obtained. Afterwards, the resulting sequence is compared against a reference sequence in order to obtain the differences between them. Geneticists name these differences as genetic variations and use them to assess their effects in the humans' health. Nowadays, it is possible to predict the risk of getting a certain disease by searching for specific genetic variations on the DNA sequence [1].

Geneticists perform DNA sequence analysis aided by bioinformatic tools. Even though these tools are functional and useful for reducing time and complexity, none of them completely fulfill all the geneticists' requirements [2]. As a consequence, geneticists are forced to use several tools in order to gather all the functionality and, eventually, accomplish the complete DNA sequence analysis.

One important issue regarding these tools is that data exchange among them is required. The problem lies in the fact that each of these tools is isolated and uses its own data format to report the computed information. For this reason, data exchange among tools is a non-trivial task that geneticists must address in each analysis. An example procedure to fulfill this task is described as follows:

1. Data exportation: After the tool task is completed, the results are exported into a file following one of the available export formats.
2. Format comprehension: It is required to understand the semantics and the syntax of the tool-specific data formats. The concepts related with the information that has to be exchanged should be identified in both involved formats.
3. Format manipulation: A relation between both formats is established for each required concept. Then, all the data is translated from the source format into the target format.
4. Data importation: Once all the information is expressed using one of the available import formats, it can be imported in the target tool.

As geneticists usually lack Software Engineering knowledge, most of them perform this task manually or by means of programming scripts. Although these specific scripts are useful in solving minor problems, they are far from being compliant with good practices of Software Engineering. The implemented scripts to support data exchange are often coupled solutions that integrate only two specific tools. In the end, these solutions cannot be reused and compromise the geneticists flexibility for using other tools.

In order to achieve a higher effectiveness in the execution of DNA sequence analysis tasks, the availability of structured genomic information systems, and software tools to exploit them, is very important. The work presented in this paper is part of the Diagen Project, a research project created to address these goals. The Diagen Project is a collaborative project among experts of different domains: 1) Geneticists from the Instituto de Médicina Génomica (IMEGEN), experts in DNA sequence analysis; 2) Software Engineers from the Centro de Investigación en Métodos de Producción de Software (ProS), experts in the application of Model-Driven Software Development in different domains; and 3) Computer Scientists from the Grid y Computación de Altas Prestaciones (GryCap), experts in parallel computation and supercomputation.

The first objective of the project (Figure 1) was the design of the Conceptual Schema of the Human Genome (CSHG) [3], a conceptual schema whose main aim is the formalization of the concepts related with the human genome.

The second objective of the project was the development of an Information System based on the CSHG, the Human Genome Database (HGDB). This database has been populated with structured genomic information by means of loading routines that understand, transform and load the genetic data obtained from different heterogeneous genomic databases.

The last objective of the project, explained in this paper, was the identification of the effectiveness problems of current software tools available for DNA sequence

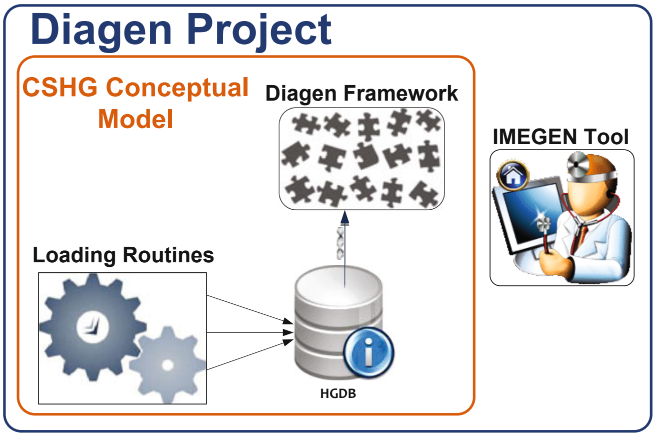


Fig. 1. Diagen Research Project

analysis and the elaboration of a tool that overcomes these problems. This study revealed that the problems were not in the tools functionality itself, but in the lack of support for data exchange among them.

As a solution, this paper proposes the application of conceptual modeling to develop a model-driven framework that formalizes data exchange and simplifies tool integration. The Diagen framework gathers the required conceptual models with the aim to establish a common domain specification for expressing genetic data precisely and unambiguously. Moreover, it provides a suitable mechanism to manage data flow among tools to achieve the tool integration. As a proof of concept, the proposed framework integrates several tools that are used by IMEGEN geneticists in their daily routine to search for genetic variations, using real DNA samples of the BRCA2 gene (a gene related to Breast Cancer).

The rest of the paper is organized as follows: Section 2 presents a brief summary of other proposed solutions to solve the tool integration problems in DNA sequence analysis. Section 3 explains the proposed model-driven framework for integrating bioinformatic tools. Section 4 presents how the framework is used for disease diagnosis support using samples of the gene BRCA2 and a set of bioinformatic tools. And finally, section 5 presents the conclusions and future work.

2 Related Work

Several works have attempted to overcome current DNA sequence analysis tool issues. These proposals follow two different approaches.

Several sequence file formats for expressing bioinformatic tools results have emerged. Examples of these formats are: 1) Variant calling formats, such as the Variant Call Format (VCF) proposed for the 1000 Genomes Project [4]; and,

2) Alignment results formats, such as the Genome Variation Format (GVF) [5], which provides a textual format using the Sequence Ontology [6] or the Sequence Alignment/Map Format (SAM) [7], which provides a compressed textual representation of read alignments against a reference. The SAM authors already proposed a set of utilities, named SAM tools, to provide post-procesing functionality to the SAM format (such as viewers). The SAM format expresses large amounts of alignments in small size and allows an efficient access to them. However, the interpretation of this format is not easy and requires a good knowledge of the format in order to identify the configuration and codification of their different fields.

All these formats have been defined for the purpose to provide interoperability among different DNA sequence analysis tools. The implementation of decoupled data exchange mechanisms is feasible using any of the above examples as a standard format. However, their main drawbacks are the complexity of each textual format and the mandatory implementation of a low-level mechanism to extract the data. As a consequence, none of them have become a widely applied standard and are only used in the research context where they have been proposed.

Several bioinformatic development frameworks have also been implemented that address the integration issue. Some examples of these frameworks are Biojava [8], BioPython [9], or BioPerl [10]. These frameworks provide an API that supports common functionality for DNA analysis tasks. These frameworks have been defined to provide geneticists with the freedom to implement their personalized tools. All of them allow geneticists to develop programs written in different programing languages (Java, Python or Perl) offering implemented methods that can be called inside their programs. Additionally, as the sequencing domain lacks of standard nomenclature to express the output results, they provide several format conversion operations to transform file formats among different tools. However, although geneticists are able to customize their programs for DNA sequence analysis, they still have to worry about low-level programming details and integration issues.

Another example of a development framework is the Taverna Tool [11], a framework for the design, edition and execution of workflows based on the integration of web services. Taverna is specially focused on the biological domain, providing the interoperability with biological resources such as myExperiment or the BioCatalogue. The Taverna tool is a very well-known tool among biologist to design in-silico experiments. However, geneticists still have to worry about the inputs and outputs that each web service produces, that is, they must map which output is required as input for the next task.

Concerning the academic environment, some approaches have considered the use of conceptual modeling to improve the quality of software tools for DNA sequence analysis. On the one hand, the framework Pierre [12] is used for the partial generation of user interfaces for browsing through genomic repositories.This automatic generation is based on the specification and composition of different genomic services. On the other hand, the framework MEMOPS [13] presents an architecture for the retrieval and storage of biological information. This framework uses UML as modeling language in order to define a data model that is used to generate automatically the documentation, the programing interfaces and the storage code. Both approaches follow model-driven principles in order to create usable interfaces, but they addressed how to present and retrieve genomic information, instead of processing tasks to obtain additional analytical information or integration with another bioinformatic tools.

The Diagen framework allows the integration of bioinformatic tools in order to design computational processes to obtain new genetic knowledge from their data. Moreover, the framework is designed to implement high-level services to avoid specific tasks and input/output formats issues.

3 An Integrative Framework for Bioinformatics

This work presents a model-driven framework for the integration of DNA sequence analysis tools and the retrieval of genetic information. Diagen is classified as a model-driven framework because each of its components (classes, data entities, and operations) is a projection of the Conceptual Schema of the Human Genome (CSHG).

The CSHG is a conceptual model created in close collaboration with geneticists, where biological concepts related to the human genome have been precisely addressed and defined. The CSHG has been designed through the definition of four views: 1) The structural view, which defines the internal composition of different genetic elements; 2) The variation view, which defines the knowledge related to the differences in DNA sequences among individuals; 3) The transcription view, which defines the functional effects through the production of proteins; and 4) The pathway view, which defines the metabolic reactions that occur inside the cell. These views contain the concepts related to different perspectives of the human genome properties.

The Diagen framework uses three views (structural, transcription and variation) of this conceptual model to support the following DNA sequence analysis tasks (Figure 2):

1. Sequence Treatment (T1): Due to technological limitations of sequencing techniques, the process is fragmented and error prone. As a consequence, the final DNA sequence is made up of small and redundant fragments that have to be assembled by a specific program using a reference sequence; eventually a consensus sequence is derived.
2. Sequence Alignment (T2): The resulting DNA sequence is aligned to a reference sequence in order to determine the differences between them. Each difference is classified according to the change that has occurred in the DNA sequence.
3. Variation Knowledge (T3): Each difference is characterized as a genetic variation. Moreover, using data gathered in genomic databases, each variation is associated with complementary information and reported if it is associated to a disease.

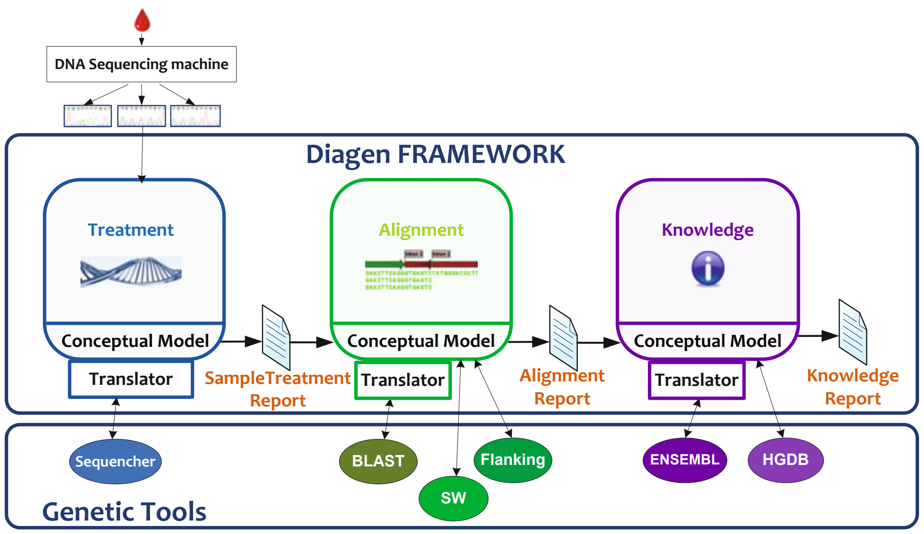


Fig. 2. General View of the Framework

In order to accomplish a complete DNA sequence analysis, these tasks must be executed sequentially and exchange information among them. Taking into account that data exchange is required when a tool calculates data that another tool requires, it can be assumed that both tools must share a set of common concepts. Therefore, it is possible to define a conceptual model that represents those shared concepts and establishes well-defined boundaries and vocabularies. These conceptual models are called *Reports* in the framework context, as they gather all the concepts related to the information that is reported in each task.

Diagen establishes a common context to guide data exchange among tools defining a conceptual model for each task transition. Focusing on the DNA sequence analysis process, three conceptual models have been defined:

1. The Sample Treatment Report conceptual model (Figure 3): This conceptual model defines all the concepts related to the reconstructed sequence obtained after the sequence treatment task (T1). This sequence is analyzed in the sequence alignment task (T2).
   The entity *Gene* represents the DNA region that is sequenced by geneticists. A *Gene* is identified by the *id*, that is a well known term in the genetics community. The attribute *transcript_id* identifies the transcription sequence that has been used to determine the set of exons to be sequenced. On the one hand, a *Gene* has a set of *Exons*, the regions of the genes that codify for proteins. An *Exon* is identified by a number *num*, that locates its position inside the *Gene*. The attribute *consensus* contains the validated DNA sequence of the *Exon* after the sample treatment process. On the other hand, a *Gene* is made up by *Segments* that are the fragments obtained by the sequencing machines. A *Segment* is identified by the attribute *id*, that differ-

entiates one from another. The attribute *electropherogram* contains the result obtained directly from sequencing machines and the attribute *sequence* contains the interpretation of this *electropherogram* into nucleotides (A,T,G,C). The *startpos* and *endpos* indicate the range of the fragment inside the *Gene*. Finally, the *Reference* entity represents the sequence used by geneticists as a reference to carry out different analysis. In this task, the *Reference* is used for the assembly of all the *Segments* of the *Gene* in order to obtain the *consensus* sequences. The attribute *id* identifies the entity using a well known nomenclature in the genetics community. The attribute *sequence* contains a general sequence, that is a computed representation of all the sequences of the human beings. This conceptual model uses a few concepts of the structural and transcription views, such as the *Gene* entity, the *Exon* entity, or the *transcript_id* attribute. Other concepts are related with technological details, such as the *Segment* entity or the attribute *electropherogram*.

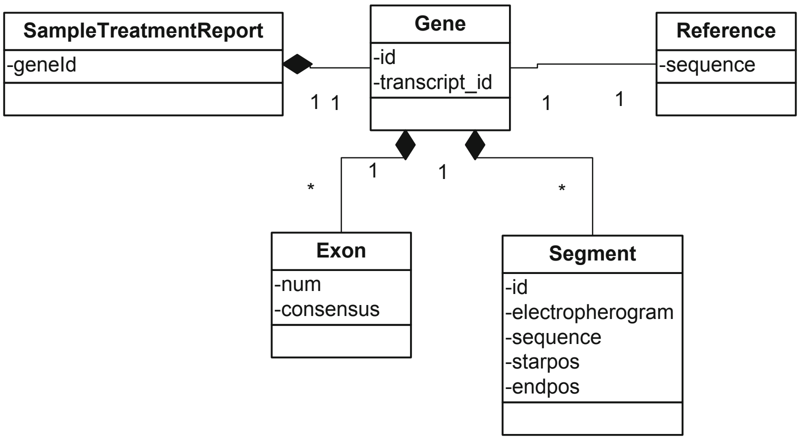


Fig. 3. Sample Treatment Report Conceptual Model

2. The Alignment Report conceptual model (introduced in [14], Figure 4): This conceptual model defines all the concepts related to the differences found in the sequence alignment task (T2) to be characterized in the variation knowledge task (T3).
   Gene and *Reference* are the entities that are compared in the Alignment task. A *Gene* represents the DNA sequence that geneticists want to analyze, and is the generalization of the Gene and Exon entities from the Sample Treatment conceptual model. The *Reference* is the reference sequence used for comparison and is the same as the one described in the Sample Treatment conceptual model. As geneticists are only interested in differences, the Alignment Report contains a list of the *Differences* found. A *Difference* is identified by *startPos* and *endPos* that locates it according to the *Reference*' sequence. Moreover, a *Difference* may be identified as well using their

flanking sequences: 20 nucleotides that are delimiting both sides of the difference, the *fsright* and *fsleft* attributes. The attribute *isHeterozygous* is a biological concept that indicates if the difference has occurred in one (homozygous) or both DNA strands (heterozygous). A *Difference* is categorized in three types: *Insertion*, *Deletion* or *Substitution* if some *characters* have been introduced, deleted or substituted in the sequence in comparison with the reference. This conceptual model uses a few concepts of the structural view, such as the entities *Gene* and *Reference*. Other concepts are related with string sequences comparison, such as the entities *Insertion*, *Deletion* or *Substitution*, and the attribute *characters*.

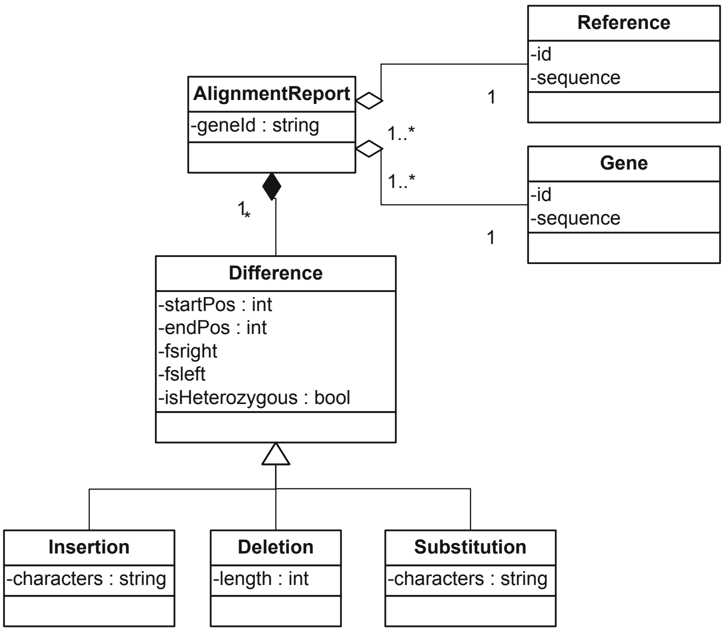


Fig. 4. Alignment Report Conceptual Model

3. The Knowledge Report conceptual model (Figure 5): This conceptual model defines all the concepts related to the characterized variations in the knowledge task (T3) to be used for another task, for example, to visualize a diagnosis report.

   Reference and *Gene* are the entities that have been analyzed in order to create a disease diagnosis report. From this analysis a list of genetic variations have been detected. Each difference described in the Alignment conceptual model becomes a *Variation* when it has some genetic knowledge associated. A *Variation* is characterized by the same attributes than the differences: *startPos* and *endPos* according to the *Reference*; the *fsright* and the *fsleft* flanking sequences of the *Variation*; and the *isHeterozygous* attribute indicating the affected strands of the variation. Additionally, a *Variation* is characterized by the attributes *HGVSGenomic*, *HGVSCoding* and *HGVSProtein* that represent the standard nomenclature for expressing genetic variations [15]. A *Variation* can be categorized in three types: *Insertion*, *Deletion* or *Indel* if some *nucleotides* have been inserted, deleted or changed. The entity *Knowledge* retrieves the additional information that has been found in genetic databases.The attribute *phenotype* expresses the external feature associated to the variation. The attribute *isSNP* means that is a common variation present in at least 2% of the population. The attribute *certainty* offers an assessment of the veracity of the information. Finally, the *source* refers to the data origin. Sometimes, this information is claimed to be true by a research publication. The entity *Bibliography* shows the information about this publication: the *title* of the paper, the *authors* involved, the *abstract* or summary of the contents, the media of *publication* (for example, a conference or a journal), and the *URL* where it can be found. This conceptual model uses a few concepts of the structural view, such as the entities *Gene* and *Reference*; the variation view, such as the *Variation* entity, or the attributes *phenotype* and *isSNP*; and the transcription view, such as the attributes *HGVScoding* and *HGVSprotein*.

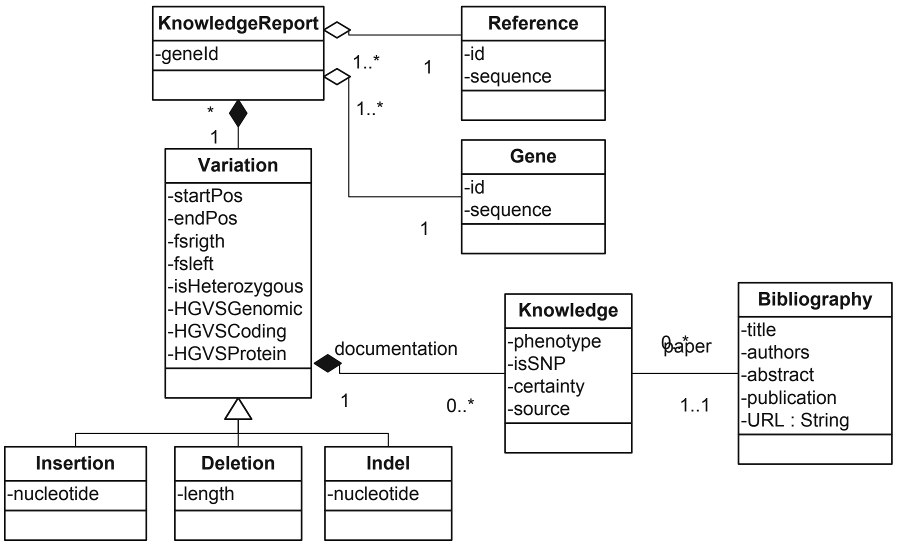


Fig. 5. Variation Knowledge Conceptual Model

Data exchange among tools that perform these tasks, usually requires the implementation of a translation mechanism to understand each other. In that case, data expressed in a concrete format needs to be translated into a different format. The implementation of this translation mechanism is a highly consuming task, and the solution is not flexible enough if some tool has to be introduced or changed in the process.

Instead, Diagen avoids coupled implementations thanks to the use of conceptual models, in a higher level of abstraction. A tool can be completely integrated in the process incorporating a simple translator in the framework. This translator is easier to implement since it only requires establishing the relationships between the output and the conceptual model. This translator should express the outputs of the tool in terms of the underlying conceptual model. Hence, the implementation of this translator is completely independent of other tools and formats.

The Diagen framework has been implemented using the Java language. Additionally, each conceptual model involved in data exchange has software correspondence with a set of Java classes and a XML representation. In order to manage both representations (Java and XML) JAXB (Java Architecture for XML bindings) [16] has been used. This is a specific API that allows Java objects to be parsed in XML data and vice versa. The implementation of both configurations provides a better flexibility to use each of the components into another environments.

Each task that is supported by the framework has been implemented to be independent from the others, therefore it can be used separately. Thanks to this modularity, it is possible, for example, to use the alignment task in another environment (Figure 6). In this case, the input data must be provided in terms of the input conceptual model (Sample Treatment Report) and the output report must be read in terms of the output conceptual model (Alignment Report). Both conceptual models can be expressed using the Java language or the XML representation. In the case of XML, the JAXB framework manages to the correspondence with the Java language.

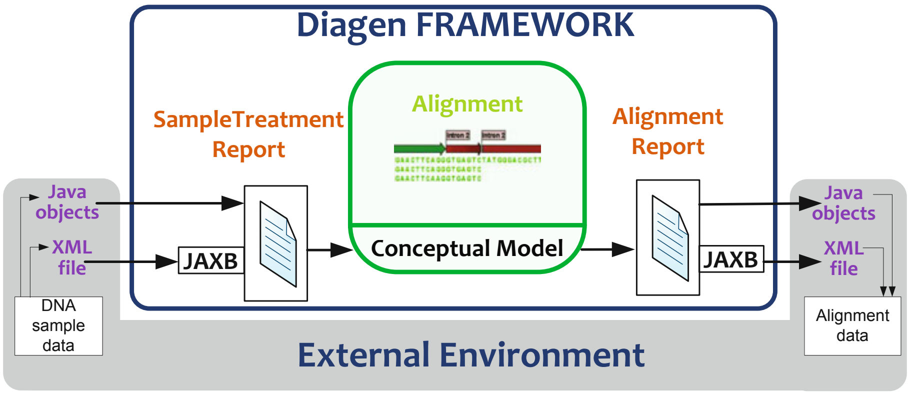


Fig. 6. Integration Mechanism for Alignment Task

4 Using Diagen for Disease Diagnosis: The BRCA2 Case

As a proof of concept, the framework has been used to develop a prototype for disease diagnosis of Breast Cancer. The prototype has been designed as a web application in order to offer geneticists a higher flexibility and avoid them installation issues. This specific framework configuration integrates several bioinformatic tools that are used daily by the geneticists of IMEGEN.

The prototype architecture consists of four layers (Figure 7):

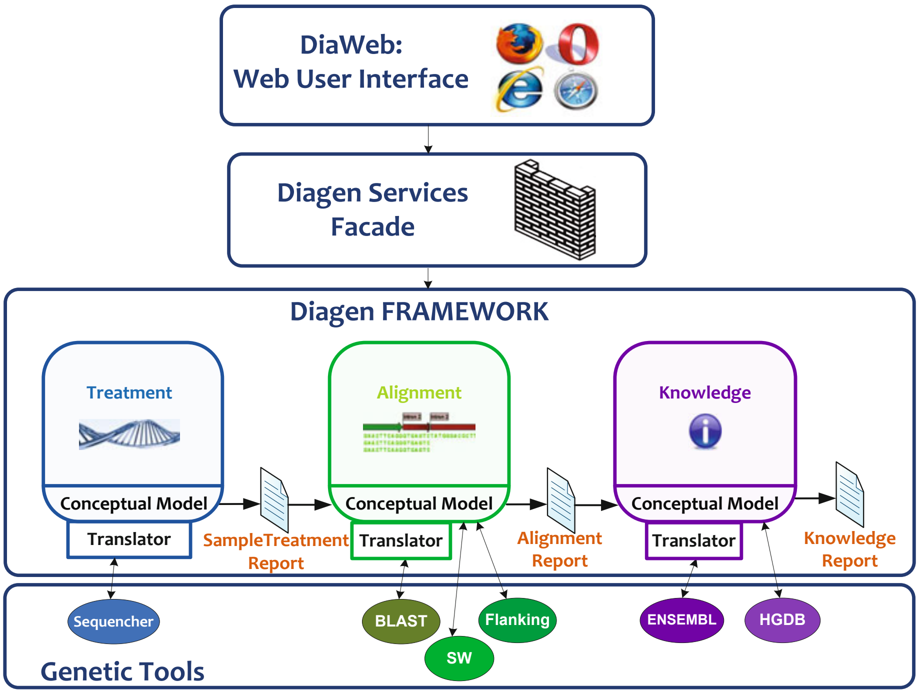


Fig. 7. IMEGEN Prototype using Diagen

1. DiaWeb: This layer is responsible of the interaction with geneticists through several web pages that capture data (such as the sequencing machine output or the selection of the algorithm for aligning the sample) and offer a diagnosis result.
2. Diagen Facade: This layer offers to upper layers the Diagen functionality through an API. This interface is defined as several methods that encapsulate the available services of Diagen.
3. Diagen Framework: This layer includes the presented framework and the tool translators. These translators are responsible of translating the tool format into the conceptual model, and vice versa.

4. Genetic Tools: This layer collects the tools that geneticists from the Diagen research project usually use in order to accomplish the DNA sequence analysis. It also includes some additional tools designed in the context of this project with the aim of improving the efficiency of these tasks. Hence, the framework has been applied to integrate:
   (a) Sequence treatment task: The Sequencher tool [17] is used to rebuild the sequence from the segments provided by sequencing machines. A translator, which expresses the calculated consensus sequences in terms of the Sample Treatment Report conceptual model, has been implemented and incorporated inside the framework.
   (b) Sequence Alignment task: The implementation of the BLAST algorithm from NCBI [18] is used to search for differences in the sequence. In this case, the incorporated translator expresses the found differences in terms of the Alignment Report conceptual model. Due to the fact that BLAST is an heuristic algorithm, it does not always detect the best solution to the alignment problem. Hence, an algorithm based on the Smith-Watterman has been implemented (SW Tool). This algorithm, although is more accurate than BLAST, it is still not efficient enough to be used in practice. For these reasons, another alignment approach has been proposed: to look for known-variations in the sequence by aligning the flanking sequences of each variation against the DNA sequence (Flanking Tool). Regarding both tools, the integration does not required the implementation of an additional translator because both use the proposed conceptual model and their results are already expressed using the Alignment Report conceptual model.
   (c) Variation Knowledge task: Variation characterization is performed manually by geneticists searching in several databases. As a better solution, the Diagen framework integrates two mechanisms for genetic knowledge retrieval. The first mechanism obtains some genetic data (such as structural information about genes or transcription data required to calculate the HGVS notation) from the ENSEMBL database [19]. In this case, a translator has been included to express this data using the conceptual model. However, the information retrieved by ENSEMBL is not enough to execute the complete variation knowledge task. The second mechanism retrieves genetic information (such as phenotype information or bibliographic references) from the HGDB database [20]. As the HGDB gathers information from different genomic repositories that geneticists usually check, it is possible to execute the variation knowledge task successfully. Moreover, as HGDB is based on the Conceptual Schema of the Human Genome (CSHG) [3] it does not require an additional translator.

The prototype supports the three defined tasks to perform a DNA sequence analysis. As a result, it retrieves a personalized report containing the genetic variations and the potential diseases according to an individual sample.

The use of the framework provides two main advantages: 1) The development time of specific translators decreases because the core functionality can be reused among them; 2) The framework also provides common functionality for managing DNA sequences (comparison, retrieval of reference sequences, nomenclature and so on). Moreover, regarding the use of the IMEGEN tool itself the main advantages are: 1) A reduction in the efforts needed for data exchange among tools, as the Diagen framework manages data flow transparently for geneticists; 2) The elimination of the need to search for variation data in the huge set of databases spread around the Web, as the information system HGDB gathers this data in a structured way and the genetic data retrieval is easily performed; and 3) A decrease in the execution time, as manual data flow and manual search are avoided.

The prototype has been tested with real samples of the BRCA2 gene (Table 1). The test was carried out analyzing the BRCA2 gene sample from ten different patients (P1-P10). For each patient, the table shows the number of variations characterized by IMEGEN, the number of variations characterized by Diagen, and the accuracy that Diagen offers compared with the IMEGEN manual process. IMEGEN performs the analysis in approximately four hours (depending on the success achieved while searching for a difference in the genetic repositories).

The preliminary test showed that Diagen offers the results almost instantly and with an accuracy rate of between 60-90%. It is also important to emphasize that the variations not characterized by Diagen were always the same variations (7 variations in total) that appeared repeatedly in all the analyses.

Table 1. Preliminary BRCA2 tests

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Characterized Var. IMEGEN	7	10	8	8	8	13	9	10	9	8
Characterized Var. Diagen	6	6	7	6	5	8	6	7	6	5
Accuracy rate %	86	60	88	75	63	62	67	70	67	63

5 Conclusions and Future Work

This work proposes a model-driven framework that is based on a well-defined conceptual model of the human genome in order to address DNA sequence analysis. As a proof of concept, the Diagen framework is configured for the development of a disease diagnosis support and is tested by means of real DNA samples of the BRCA2 gene.

We have realized that the available tools actually accomplish some of the geneticists' goals. The problem lies in the fact that geneticists' activities, specifically in the DNA analysis domain, lack standard methodologies, well-defined processes, fixed vocabularies, and unified knowledge sources. As a consequence, the execution of a DNA sequence analysis cannot be performed efficiently or without geneticists' intervention.

The solution to these problems is not to reinvent new DNA sequence analysis tools but to integrate the most suitable tools according to geneticists' needs. The presented framework applies conceptual modeling to integrate different bioinformatic tools and to provide a common context to exchange data with each other. The main advantage of the presented framework, over other integration approaches, is that Diagen is a high-level abstraction framework that provides concise and significant tasks to geneticists (such as "sequence treatment", "alignment" or "variation knowledge retrieval") instead of low-level tasks (such as "run Blast algorithm", "translate format to Fasta", or "obtain HGVS nomenclature"). Moreover, with this framework, geneticists can perform a DNA sequence analysis and forget about the data formats of different tools.

As genetics is a very innovative field that is constantly evolving with new discoveries, all concepts must be well-defined without ambiguity. The CSHG was the first step in order to establish the domain specification of the different human genome concepts. Thanks to this conceptual schema is possible to specify the concepts related to DNA sequence analysis. Thus, the involved concepts in the different tasks have been precisely formalized. As a consequence, it is easier to adapt the tasks to changes or to support new concepts.

The preliminary results are promising, but there is room for improvement. The low accuracy detected is because the missed variations were not described in the integrated sources. As these sources are constantly improving, it is expected that future versions of the IMEGEN prototype will solve these issues.

As future work, the framework will be extended to support other bioinformatic tasks. The main goal of this extension is to design a complete framework that supports other functionality besides DNA sequence variation analysis.

Additionally, the next step is to apply the service-oriented paradigm to provide a more flexible development environment. With this approach, geneticists could select only the required functionality, defined as services, and easily create a personalized tool.

Acknowledgments. Thanks to the Instituto de Medicina Genómica (IMEGEN, http://www.imegen.es) for its support and the provision of data. This work has been developed with the support of MICINN under the FPU grant AP2010-1985 and the project PROS-Req (TIN2010-19130-C02-02), and co-financed with ERDF.

References

1. Hamburg, M.A., Collins, F.S.: The Path to Personalized Medicine. New England Journal of Medicine 363(4), 301–304 (2010)
2. Rusk, N.: Focus on Next-Generation Sequencing Data Analysis. Nature Methods 6(11s), S1 (2009)
3. Pastor, O., Levin, A.M., Celma, M., Casamayor, J.C., Virrueta, A., Eraso, L.E.: Model-Based Engineering Applied to the Interpretation of the Human Genome. In: Kaschek, R., Delcambre, L. (eds.) The Evolution of Conceptual Modeling. LNCS, vol. 6520, pp. 306–330. Springer, Heidelberg (2011)
4. Nayanah, S.: 1000 Genomes Project. Nat. Biotech. 26(3), 256 (2008)

5. Reese, M.G., et al.: A Standard Variation File Format for Human Genome Sequences. Genome Biology 11(8), R88 (2010)
6. Eilbeck, K., et al.: The Sequence Ontology: A Tool for the Unification of Genome Annotations. Genome Biology 6(5), R44 (2005)
7. Li, H., et al.: The Sequence Alignment/Map Format and SAMtools. Bioinformatics 25(16), 2078–2079 (2009)
8. Holland, R.C.G., et al.: BioJava: An Open-Source Framework for Bioinformatics. Bioinformatics 24(18), 2096–2097 (2008)
9. Cock, P., et al.: Biopython: Freely Available Python Tools for Computational Molecular Biology and Bioinformatics. Bioinformatics 25(11), 1422–1423 (2009)
10. Stajich, J.E., et al.: The Bioperl Toolkit: Perl Modules for the Life Sciences. Genome Research 12(10), 1611–1618 (2002)
11. Hull, D., et al.: Taverna: a tool for building and running workflows of services. Nucleic Acids Research 34(Web Server issue), 729–732 (2006)
12. Garwood, K., et al.: Model-driven user interfaces for bioinformatics data resources: regenerating the wheel as an alternative to reinventing it. BMC Bioinformatics 7, 532 (2006)
13. Fogh, R.H., et al.: MEMOPS: Data modelling and automatic code generation. Journal of Integrative Bioinformatics 7 (2010)
14. Villanueva, M.J., Valverde, F., Pastor, O.: Applying Conceptual Modeling to Alignment Tools: One Step towards the Automation of DNA Sequence Analysis. Bioinformatics 2011 (2011)
15. Dunnen, J.T., Antonarakis, S.E.: Mutation nomenclature extensions and suggestions to describe complex mutations: A discussion. Human Mutation 15, 7–12 (2000)
16. Ort, E., Mehta, B.: Java Architecture for XML Binding (JAXB). Technical Report Sun Developer Network (2003)
17. Curtis, P., Bromberg, C., Cash, H., Goebel, J.C.: Sequencher, Gene Codes Corporation, Ann Arbor, Michigan (1995)
18. NCBI BLAST (Basic Local Alignment Search Tool), http://blast.ncbi.nlm.nih.gov
19. Hubbard, T., et al.: The ENSEMBL Genome Database Project. Nucleic Acids Research 30(1), 38–41 (2002)
20. Pastor, O., et al.: Enforcing Conceptual Modeling to Improve the Understanding of Human Genome. In: Research Challenges in Information Science (RCIS), pp. 85–92 (2010)

A Software Framework for the Automated Production of Schematic Maps

Joao Mourinho[1], Teresa Galvao[1], João Falcão e Cunha[1], Fernando Vieira[2], and Jose Pacheco[2]

[1] Faculdade de Engenharia da Universidade do Porto, 4200-465 Porto
[2] OPT - Optimizacao e Planeamento de Transportes, 4200 Porto
joaomourinho@fe.up.pt

Abstract. Schematic Maps are mainly used for depicting transportation networks. They are generated through a schematization process where irrelevant details are eliminated and important details are emphasized. This process, being manually performed by teams of expert designers, is expensive and time consuming. Such manual execution is unsuitable for the production of schematic maps for location-based services or on-demand schematic maps, as near real-time and user-centered properties are needed. This work proposes GeneX, a framework that can support the automated generation of schematic maps. The framework and a new algorithms developed were able to completely eliminate erroneous map point placement, and to decrease by 33% the contention for map point placement, producing schematic maps without human intervention in soft real time.

Keywords: Schematic Maps, Software Framework, Public Transportation.

1 Introduction

Maps are part of the communicative process intended to communicate space information: the producer of maps (the sender) communicates to the receiver the message (map). The making of maps was only possible through the use of symbols and abstraction (which serve as a language), as maps are mainly intended to communicate space information. Schematic maps have been increasingly used in response to the need of better and simpler maps to describe complex transport networks, and they take the abstraction of space to a higher level, reducing the amount of information in comparison with traditional maps. This apparent simplicity is achieved through a simplification process called "schematization process" where choices are made regarding the level of detail and simplification. A special type of schematic map, called spider map, has also appeared recently. It presents innovative features such as a spider structure to improve visual presentation, user learning and spatial context communication. Schematic maps, by their inherent simplicity and symbolic meaning are good maps for being used in the transportation area as they are far more intuitive than conventional maps [1].

In fact as people travel more often, they need flexible and easy to understand maps which may take in account their context [2]. Automation in the production of maps is a key factor to achieve flexibility to tailor maps to user context, as it happens with Location-Based Services [3] where "information services accessible with mobile devices through the mobile network which have the ability to make use of the location of the mobile device" [4]. There is the need, then, to develop a software framework which could support efficiently and comprehensively the automated generation and test of schematic maps. In this paper we propose and describe a software framework which serves as an engine to the automated generation and test of schematic maps. The structure of the paper is as follows. The main concepts regarding the definition of a schematic map and in particular, of a special type of context based schematic maps that we have called Spider Maps, are presented in Section 2. In Section 3 we present the related work concerning the schematization algorithms. In Section 4 we present the framework and discuss its development, architecture, and its enhancements. In section 5 we discuss the results obtained so far. Section 6 provides concluding remarks and clues about future work.

2 Spider Maps

2.1 Schematic Maps

Since the begining of the industrial revolution, the need of highly efficient, easily understandable transportation maps pushed the evolution of the traditional maps, and new forms of cartographic representation have emerged. Among the new forms of cartographic representation that have emerged, the schematic maps were probably the most bold. One remarkable schematic map applied to a transportation network was the Harry Beck's London Underground diagram, depicted in figure 2. As it is possible to observe, it presents radical changes with former London underground maps presented in figure 4.

Beck's map was considered both bold and innovative, as for the first time lines were drawn either horizontally, vertically or diagonally at 45. This map also uses a non-linear scale, so the central area of the diagram is shown at a larger scale than the extremities. It shall be noted that although it does not mimic the geography of London, this map gives the traveler some clues about the terrain features (ex: river) and his/her location. Some authors define schematic map as *"an easy-to-follow diagrammatic representation based on highly generalized lines which is in general used for showing routes of transportation systems, such as subways, trams and buses, or for any scenario in which streams of objects at nodes in a network play a role"* [5]. The most important advantage of schematic maps is that they provide a quick view of the layout of the network by removing unimportant information like the detailed shape of the connections.

2.2 Spider Maps

Schematic spider maps are a special type of schematic maps. As with traditional schematic maps, the stops and lines of the transportation network correspond to

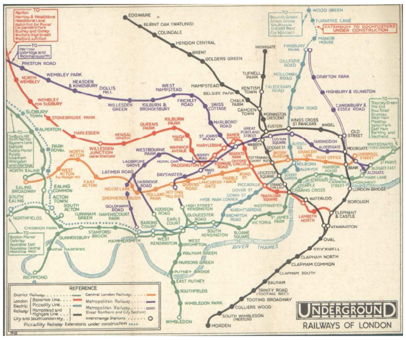


Fig. 1. London Underground Map prior to Beck's map. *Source: http://www.chadsayshello.com*

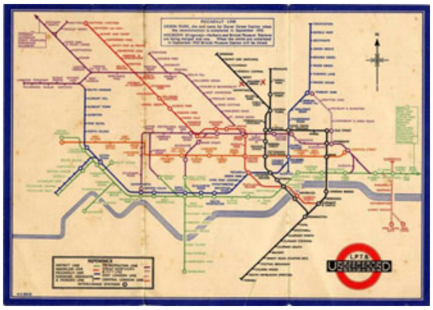


Fig. 2. Harry Beck's London underground map. *Source: http://britton.disted.camosun.bc.ca*

vertices and edges, respectively, in the spider map. However, they have enhanced features such a spider architecture, thus having a specific set of characteristics which sets them apart from the traditional schematic maps. Spider maps pay special attention to the context in order to enhance its learning by the users and it ease of use. While a schematic map is mainly used for the representation of metropolitan networks which remain fixed for several years, a spider map can be used (in London, for example) to represent bus transportation networks which, by its nature, are much more complex and unstable. In fact, if an underground transportation network can be represented with a single schematic map, it would not be feasible to represent a relatively complex bus transportation network using a single schematic map, due to the large number of bus stops and lines. To overcome this problem, spider maps allow the depiction of a schematic map for a particular geographic area. Hence, in order to represent a bus transportation network within a city, several spider maps can be produced (similar to the one depicted in Figure [1]). The spider map architecture, depicted in figure [3], comprehends three main components:

– **Hub:** The hub is the main part of the map. Describes the area in which the user is, as well as the surrounding area with a higher degree of detail (buildings, roads, etc). The hub, as it is the central part of the spider map, is the first component the user will look at, as it makes uses of *"focus and context"* [6] and detail focusing [7] techniques. The hub is the only part of the spider map which does not comply with the 0/45/90 degrees line orientation. It can also include landmarks to allow the user to know important details about the place he currently is.
– **Lines:** The lines follow the orthogonal orientation of the traditional schematic maps, and describe the paths of the transport network where the user can go through while being at the zone depicted by the hub.
– **Stops:** The stops are the main destinations accessible to the user from the hub.

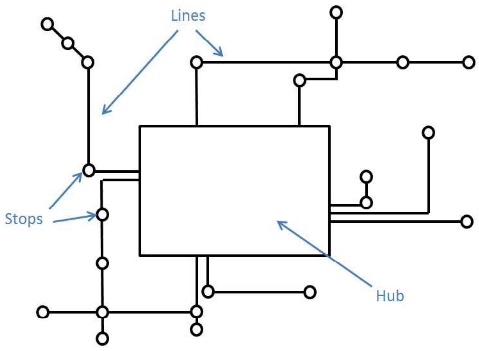


Fig. 3. Spider Map structure sketch

Figure 4 shows a spider map bus transportation network at a specific location, in this case, Victoria Station. It is possible to observe the spider map innovations in comparison with London underground maps of figures 1 and 2.

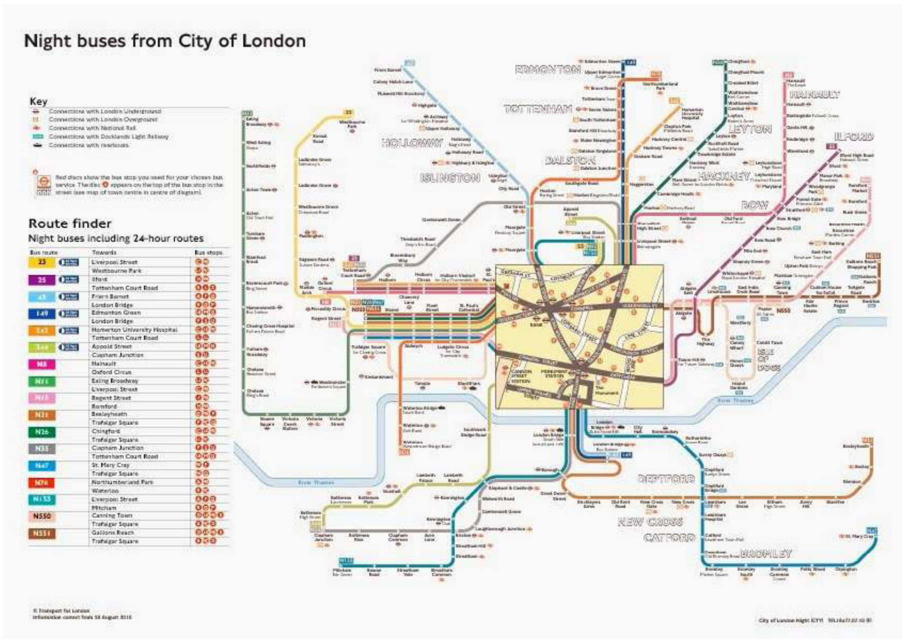


Fig. 4. London Bus Spider Map *Source: http://www.tfl.gov.uk*

3 Related Work

Schematic maps have been increasingly used in response to the need of better and simpler maps to describe complex transport networks. This visual simplicity is achieved through a sequential decision process regarding the level and nature of detail and schematization. In practice, this “schematization process” is still a manual process carried away by teams of experts, such as designers and cartographers, despite the efforts in automating the process through the use of computers. The use of computers to execute the schematization process requires effective and efficient algorithms, to achieve in one hand high quality schematic maps which can be understood by people and, in the other hand, a time efficient process. Through schematization, certain map details are emphasized while others are deemphasized. It is fundamental, however, to present the smallest amount of information the user needs to learn the map: the more information presented, the higher the learning time will be. Latto defends that information shall be reduced to its basic components to achieve that goal [7].

There are some studies regarding research on the automated drawing of schematic maps. Some studies [8] [9] focus on the rectification of lines while others focus on the optimization of schematic maps [10]. Barkowsky et al studied the application of the Discrete Curve Evolution to the schematization of maps [11]. Nevertheless these studies do not analyze the schematic map generation as a wide multidisciplinary problem as they tend to focus only on some areas of the problem. The automatic approaches for the generation of schematic maps mostly focus on the schematization process [12]. Nollenburg [13] [14] makes an extensive research on the discrete mathematical foundations which are the basis of the algorithms used in the drawing of schematic maps and makes some brief considerations about their implementation. Nevertheless, his studies do not cover the human perception factors nor a concrete computer framework for drawing schematic maps. Silvana Avelar [1] [5] presents a wider study, by including some human perception factors and makes a complete research on the schematic maps on demand, one of the components to be integrated in the new paradigm. She goes further on by presenting a framework for electronic schematic maps which can answer user queries and studies the automated generation of schematic maps. Nevertheless, the study of the human perception factors is limited to what she calls the “aesthetic factors”.

Most of the algorithms to design schematic maps retain a common structure [15]. They use a graph to model the transportation network, in which the vertices represent stops or turning points and the edges represent the paths between two turning points. In fact, the automated generation of schematic maps is truly a multidisciplinary problem, which involves integrating knowledge from several science fields. Isolated studies of different areas of knowledge, such as cognitive psychology show that user centered maps have a better performance and allow users to commit less errors [2]. Hochmair, for example [16] studied the effectiveness in the context on route planning, more specifically, as a measure of how well the map information supports the map reader in planning the fastest route between trip origin and destination on a public transportation map.

4 The GeneX Framework

In this section, we present the GeneX framework which was developed through a collaboration research performed by a team involving collaborators from FEUP[1], OPT [2], STCP[3] , FWT [4], INEGI [5], and that was funded by INEGI.

4.1 Overview

The GeneX framework is a software application designed to support the following objectives:

1. The automatic generation of electronic schematic maps for complex transportation networks in bounded time, through the flexible use and parameterization of schematization algorithms
2. To serve as a test lab to support the research of schematic maps.

By merging and processing different kinds of external information (transportation networks, geographic/topological and constraint information) through the use of state-of-the-art algorithms, the framework generates schematic maps automaticaly. It is possible for the user to choose the desired algorithm and the parameters which will influence the map schematization. The framework produces an output SVG [6] (Scalable Vectorial Graphics) file which can be used directly in an electronic way, printed in paper or further processed (e.g. manual changes). The framework may also produce a statistics file which measures several parameters about the framework functioning.

4.2 Software Engineering Life Cycle model

The requirements elicitation and validation was performed together with the project stakeholders, as part of a Joint Application Development (JAD) model [17]. The contributions provided by the partners were highly regarded: from the experts in information systems for transportation services (OPT) which provided real transport network data and expertise in automated solutions for transportation network systems, in optimization algorithms (INEGI) and map design (FWT), to the final users of the system (STCP). The development of this framework was considered, since the begining, an interdisciplinar subject in which knowledge from several areas of the science need to be integrated.

[1] Faculty of Engineering of University of Porto *http://www.fe.up.pt*

[2] OPT is an company based in Porto which develops IT infrastructures for Transportation Services. *http://www.opt.pt*

[3] STCP is a public transportation company operating in Porto. *http://www.stcp.pt*

[4] FWT is a company based in London which produces maps for transportation networks. *http://www.fwt.co.uk*

[5] INEGI is a research institute based in Porto.

[6] The Scalable Vectorial Graphic is XML language to describe vectorial bidimentional graphics. It is an open format created by the World Wide Web Consortium.

Regarding functional requirements, the framework should be capable of producing schematic and spider maps in a fully automated way, about any location the user may select, using several schematization algorithms. The framework should be able to obtain data through the use of a common standard protocol data schema shared by the stakeholders. Another requirement was that the producton of schematic maps should be time-bounded (by setting deadlines or iterations number), to make it able to support Location-based services. In order to serve as a test lab, the framework should allow the choice of the algorithm and its parameters. The framework should also support different schematization algorithms (genetic, linear, tabu search, GRASP, etc) and provide a common protocol to implement them. This set of functional requirements needs to be supported by a set of non-funcional requirements, which comprehends usability, performance and interoperability. Usability is fundamental as the schematic maps produced by the framework have a strong user learning component: the maps produced, as well all possible interactions should be as intuitive as possible to be quickly learned and understood by their users. The designers and the final user team members provide insightful highlights in this area. To be able to support location-based services, the framework should execute the algorithm and produce the correspondent schematic map in near real time. Therefore, the framework was designed to perform as a soft real-time system [18]. The idea of a real-time approach is to answer queries within time constraints [18]. One way of looking at a real-time system is through the metaphor of a stimulus/response system, where to each stimulus the system is supposed to produce an adequate response in an adequate time (deadline). If a specific system fails to produce a response to a stimulus until the deadline time and that means the value of the system is zero or negative (i.e. it causes cost, damage or negative profit), then that system is a hard real-time system. If a specific system fails to produce a response to a stimulus until the deadline and that means the value of the system is still positive after the deadline, then the system is a soft real-time system. This difference is depicted in figure 5.

The figure depicting the hard real-time system shows two possible value functions after the deadline time, the red line depicting a system in which the failure to meet the deadline implies losses or damages (ex: safety critical system). As it is possible to observe, if there is any kind of tolerance after the deadline (i.e. the value, although may not be at its maximum value, is still positive), we have a real-time system. This is the case with the GeneX framework. Regarding interoperability, the framework should not only be compatible with the systems and data of the stakeholders, but it also should be able to support several different algorithms (every algorithm which conforms to a common implementation interface). Therefore, a standardized transport network data specificication was implemented and a common interface for the algorithms was designed. This was achieved making use of reusable components citeNeighbors1984 [19].

The framework was developed by using C# Language, as a modern Object Oriented language which supported the requirement list.

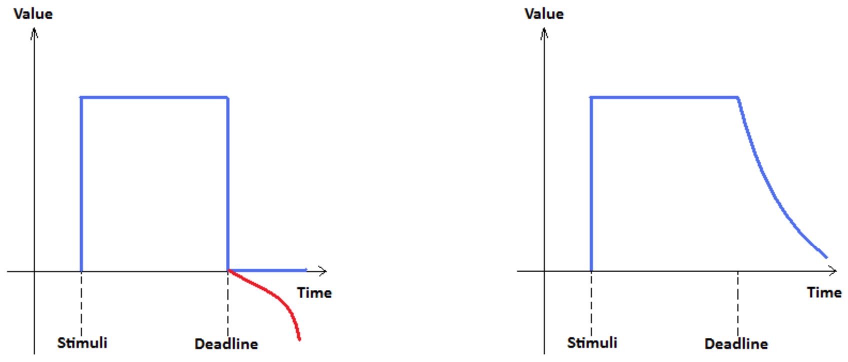


Fig. 5. Hard real-time system (left) and soft real-time system (right) stimulus/response metaphor

4.3 Architecture and Data Model

The architecture of the framework follows a modular structure, with two main modules: the data preparation module and the algorithm execution module. Figure 6 shows the GeneX framwork package diagram, depicting its components.

The data aquisition and preparation module is responsible for preparing the data to be used by the algorithm execution module. The user selects graphically the location where he wants the spider/schematic map to be centered (hub).

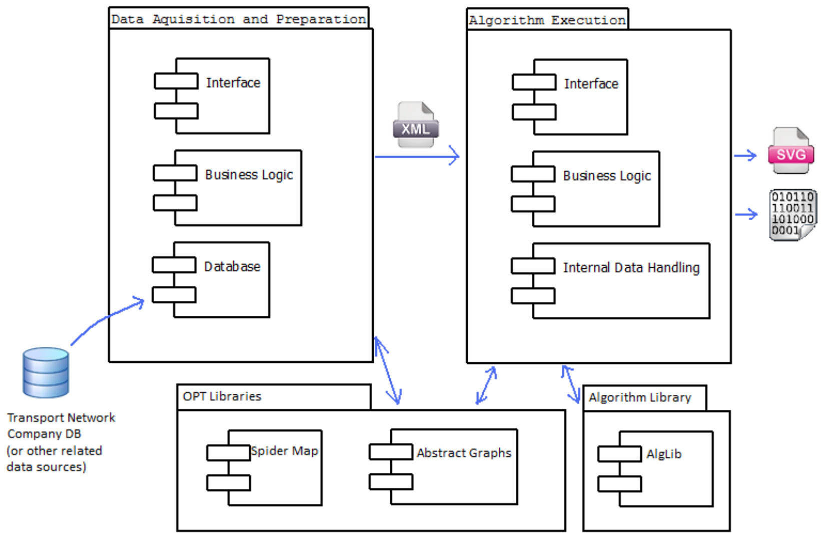


Fig. 6. GeneX Framework Package Diagram

The module then extracts raw data from the transport network database and organizes it into a data structure by using the Spider Map Library. The spider map library is a complex set of C# classes that support the serialization and communicaton of the spider and schematic map structure. Figure 7 depicts its class diagram.

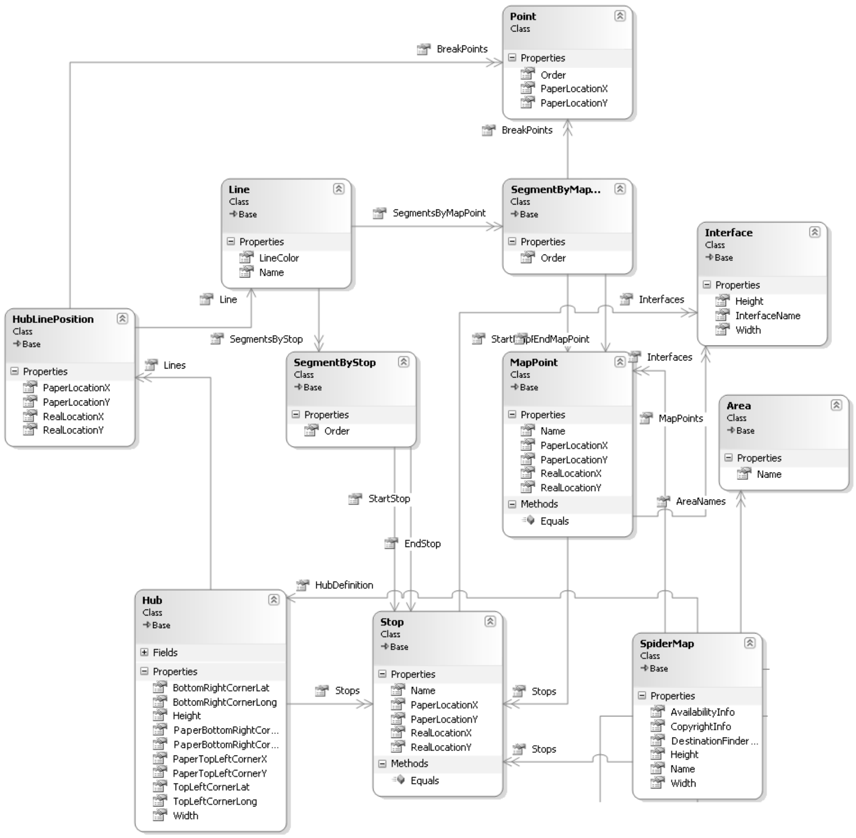


Fig. 7. Spider Map Library class diagram, showing the relations between the map components

The central class is the SpiderMap class, which contains attributes regarding the general properties of the map, such the name and dimensions. Each SpiderMap contains the definition of its hub in the Hub class. The definition of the hub encompasses its location and dimensions and also a set of HubLinePositions (which are the points where the lines start to flow from the hub to the map). The class HubLinePosition has information about its coordinates and the corresponding line (Line class). Each line may contain a set of ordered segments which have a MapPoint in each extremity. The mapPoint has location coordinates and has also a name. A line can be also a set of segments which have a Stop in each extremity. The difference between mapPoints and Stops is that a mapPoint usually depicts an certain important area made of a set of Stops. This is useful in real maps to depict very high density areas. This library is used to produce an XML file containing all the spider/schematic map data structure to be further processed by the algorithm execution module. The hierarchy of the XML file produced maps the components of the spider map according to table 1.

Table 1. Transportation Network XML Data structures mappings

Hierarchy #	Structure	Transportation Network
0	Map	Represents the whole Map.
1	Hub	The central part of the map. Useful for Spider Maps.
2	HubLinePos	The initial point of the transportation lines.
3	Line	Each line of the transportation networks.
4	Segment	Each line segment between two MapPoints.
5	MapPoints	The base element. Can represent one or a set of stops.

The algorithm execution module reads the XML file and transforms the data into an internal data structure (to improve component reuse by the schematization algorithms). At the user interface, the user can choose the schematization algorithm (available at the AlgLib algorithms package) to execute and the corresponding options and performance measurement metrics. The business logic is then responsible for calling and executing the algorithm and to produce the final result, which can be an SVG (Scalable Vector Graphics) file (a standard vectorial format) or a binary file containing serialized data. The SVG file is produced by using a library that allows the conversion of a spider map data structure in an SVG file (Abstract Graphs Library) [20].

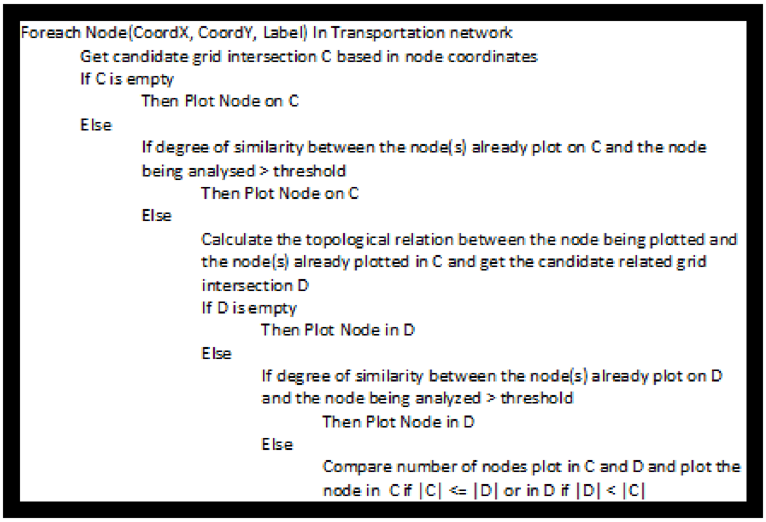
4.4 The HPPO Algorithm

The AlgLib package provides a foundation for the execution and configuration of the schematization algorithms. Each algorithm has to implement the same communication interface functions, in order to be used by the execution module:

– ***spiderMap* execute(*spiderMap*, *parameterList*)** performs the execution of the algorithm. The argumens are the spider map XML structure that was opened by the execution module, and the algorithm parameter list, already set up by the user. This function returns a spiderMap data structure which is the processed spider/schematic map. That structure can then be output to a SVG file by using Abstract Graphs library or serialized to a binary file.

– ***parameterList*** **getParameters()** the module calls this parameter function to get the list of the algorithm parameters that can be set by the user, prior to the execution of the algorithm.

We have implemented a preliminary schematization algorithm that we called "Heuristic Point Placement Optimization" (HPPO). HPPO, aligns map points (corresponding to the transportation network stops and stations) to a regular grid, by positioning each map point in the nearest grid intersection. For high density or non regular density transportation maps, map station contention for grid intersections will happen. In this case, the HPPO smartly solves the contention through an heuristic algorithm. By using regular expressions, the point labels are also taken into account when placing network vertices, by determining the similarity degree of node labels in conjunction with its geographic location in order to produce a decision about the location to plot the node where the contention arises. The use of regular expressions is based in a finite-state automaton which scans strings (sets of letters in words) in order to find the degree of similarity. Therefore, if we have a node labeled "St. James Park" and other labeled "Park St. James", the automaton will tell us that those nodes refer, with a very high degree of certainty, the same node. So we are not only relying in geographical information, but also using computer science theory and cognitive psychology (to the preservation of topological relations has positive effects in the orientation and intuitiveness [21]) to make a higher quality judgement on how to solve the contentions that arise. For each map point, HPPO starts by checking if the nearest grid intersection is empty. If it is, then there is no contention and the point is plotted there. If the grid intersection is not empty, then we have a contention. In case there is a contention, that means that the geographical coordinates of the nodes are equal, or at least, within the same decision range concerning the square grid resolution. The automaton also tells us the degree of similarity. If the degree of similarity is higher than the predefined threshold level, then we assume that both nodes refer to the same location. In this case, they should be both plotted in the same grid intersection. If the degree of similarity is lower than the threshold, then it is important to distinguish them and plot them in different grid intersections while maintaining the topological relation between them. In this case, we get the topological relation between the two points (based on their coordinates) and try to move the node to the adequate grid intersection. If contention happens again (what can happen in a highly crowded map or with a loose square grid), then we have two options: or we may continue this cycle recursively until the topological relation is violated, or we may decide if we shall plot the node into the suggested grid intersection. To limit the processing time, we discarded the recursive approach in our algorithm. Being so, we analyse the proposed grid intersection. If contention happens we check the degree of label similarity through our automaton, and if is higher than the threshold, we plot the point there. If not, we analyse both the firse and the actual grid intersections suggested and we add the node where there are less nodes plotted. The pseudoalgorithm is described in figure 8.



```
Foreach Node(CoordX, CoordY, Label) In Transportation network
        Get candidate grid intersection C based in node coordinates
        If C is empty
                Then Plot Node on C
        Else
                If degree of similarity between the node(s) already plot on C and the node
                being analysed > threshold
                        Then Plot Node on C
                Else
                        Calculate the topological relation between the node being plotted and
                        the node(s) already plotted in C and get the candidate related grid
                        intersection D
                        If D is empty
                                Then Plot Node in D
                        Else
                                If degree of similarity between the node(s) already plot on D
                                and the node being analyzed > threshold
                                        Then Plot Node in D
                                Else
                                        Compare number of nodes plot in C and D and plot the
                                        node in  C if |C| <= |D| or in D if |D| < |C|
```

Fig. 8. HPPO pseudoalgorithm description

5 Results

The GeneX Framework is able to generate in a fully automated way schematic and spider maps for every location requested. It is also a very important tool as a test lab for the schematization algorithms that are being developed. Figure 9

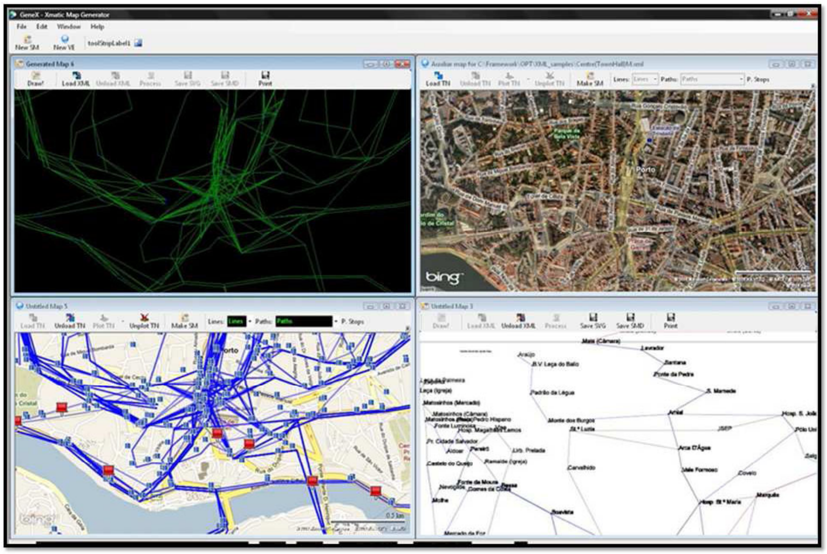


Fig. 9. User interface of the algorithm execution module, featuring an Multiple Document Interface showing several views about a schematic map

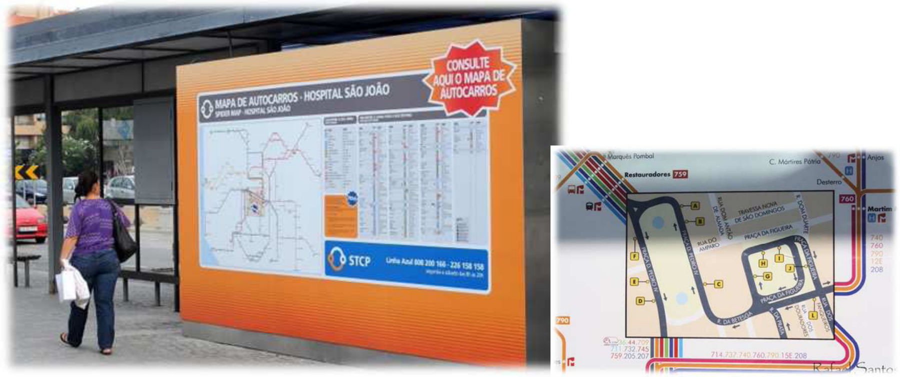
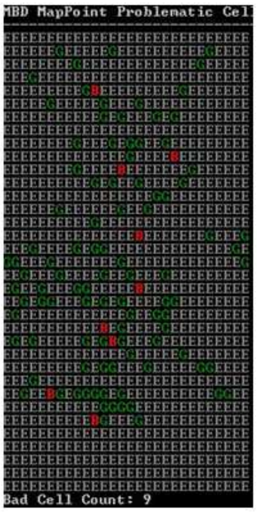


Fig. 10. Bus transportation network maps produced by using the GeneX Framework: city of Porto (left) Lisbon (right)

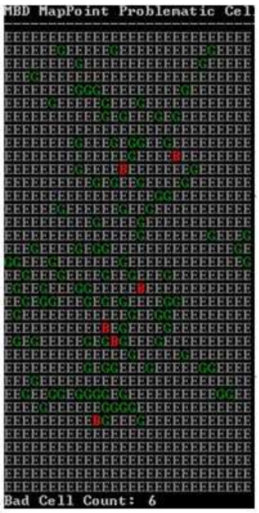
shows the user interface of the algorithm execution module, presenting multiple views of a map just before being processed by the schematization algortithms. It was found, in comparison with alternative approaches, that a multi-window environment presents a set of advantages to the generation of schematic maps, as it is much easier to assess the quality of the generated maps if we can compare side-by-side the schematization schema and the real map and geography of the terrain. This provides a valuable feedback for the tuning of the algorithms.

Although the framework is still under development and the algorithms in the AlgLib are being improved and enhanced, it is already being used for the production of schematic and spider maps. Maps produced with this framework are already available for public use in the cities of Porto and Lisbon, as it is possible to see in figure 10. As the schematization algorithms currently in use are still being perfected, those maps were also subjected to manual intervention to improve their aesthetical appearance.

Concerning our HPPO algorithm, it showed good results as can be shown at figure 11: it was shown that its use reduces by 33% the number of grid intersections that have dissimilar nodes plotted ("Bad Cells"). Other advantage of HPPO is that if different nodes refer to the same place, this algorithm can ignore contention and plots them correctly in the same grid intersection. Another advantage is that this algorithm, due to the use of regular expressions can identify nodes with similar labels which refer to the same location, grouping them in the same node. In addition, all the topological relations are still preserved. For a map with 45 MapPoints/Stops, it takes 2 seconds to execute, which is in compliance with the soft realtime requirements. Nevertheless, the decrease of 33% in the number of bad cells may not be good enough for evenly distributed very high density maps with a great number of mapPoints, as this algorithm performs better in maps which have fluctuations in mapPoint density. In the former case, its



```
MBD MapPoint Problematic Cel
Bad Cell Count: 9
```



```
MBD MapPoint Problematic Cel
Bad Cell Count: 6
```

Fig. 11. Frequency matrixes, depicting the transport network grid intersections with ASCII characters, obtained after the execution of the algorithm. It shows Bad Cells (in red) when not using HPPO(left) and using HPPO(right) for the Porto downtown spider map. HPPO reduced Bad Cells from 9 no 6, while preserving topological relations.

effectiveness can be reduced while in the later case its effectiveness can be enhanced. Therefore, this algorithm does not guarantees that the generated schematic maps contains no bad cells, so there is still margin for further improvement.

6 Conclusions and Future Work

The GeneX framework was used to produce real spider maps that are being used in the cities Porto and Lisbon, it has proved effective in generating spider maps. Altough these maps were produced by an automated process, they were subjected to small manual changes to improve the visual appearance, which is the most difficult aspect to model in the algorithms. For simpler maps with less lines and stops, the framework is able to generate them with good results. The quality of the results of the HPPO algorithm is quite good, showing a significant decrease in bad cells. The algorithms need to be perfected in order to increase the quality of the results and to make them directly usable (without any manual processing). The algorithms need to further improve aspects such as visual line distiction, stop label organization and geographical constraints. Some development of the framework is also needed to support Location-based services, such a "request manager" which can feed user requests to the framework and reply to them. Other issues that need further study are the adaptation of the resulting maps to different devices. All this work also needs to be complemented and validated with usability tests and analysis.

References

1. Avelar, S.: Schematic Maps on Demand: Design, Modeling and Visualization. PhD thesis, Swiss Federal Institute of Technology Zurich (2002)
2. Porathe, T.: User-Centered Map Design. Design (2007)
3. Steiniger, S., Neun, M., Edwardes, A.: Foundations of location based services (2006)
4. Virrantaus, K., Markkula, J., Garmash, A., Terziyan, Y., Veijalainen, J., Katanosov, A., Tirri, H.: Developing GIS-supported location-based services. In: Proc. of WGIS, pp. 423–432 (2001)
5. Avelar, S., Hurni, L.: On the Design of Schematic Transport Maps. Cartographica: The International Journal for Geographic Information and Geovisualization 41, 217–228 (2006)
6. Bogen, S., Brandes, U., Ziezold, H.: Visual Navigation with Schematic Maps. In: Visual Information Communication, pp. 65–84 (2010)
7. Latto, R.: Do we like What We See? (2004)
8. Cabello, S., Deberg, M., Vankreveld, M.: Schematization of networks. Computational Geometry 30, 223–238 (2005)
9. Cabello, S., Kreveld, M.V., Sciences, C., Box, P.O.: Schematic Networks: An Algorithm and its Implementation. In: Richardson, D., Oosterom, P. (eds.) 10th International Symposium on Spatial Data Handling (SDH), Ottawa, pp. 475–486. Springer (2002)
10. Anand, S., Avelar, S., Ware, J.M., Jackson, M.: Automated schematic map production using simulated annealing and gradient descent approaches. Technology (2000)
11. Barkowsky, T., Latecki, L.J., Richter, K.F.: Schematizing Maps: Simplification of Geographic. Cognition 8, 41–53 (2000)
12. Ware, J.M., Taylor, G.E., Thomas, N.: Automated Production of Schematic Maps for Mobile Applications. Transactions in GIS 10, 25–42 (2006)
13. Nöllenburg, M.: Automated drawing of metro maps. PhD thesis, Universitat Karlsruhe (2005)
14. Nöllenburg, M., Wolff, A.: A Mixed-Integer Program for Drawing High-Quality Metro Maps. In: Healy, P., Nikolov, N.S. (eds.) GD 2005. LNCS, vol. 3843, pp. 321–333. Springer, Heidelberg (2006)
15. Dong, W., Guo, Q., Liu, J.: Schematic road network map progressive generalization based on multiple constraints. Geo-spatial Information Science 11, 215–220 (2008)
16. Hochmair, H.: The Influence of Map Design on Route Choice from Public Transportation Maps in Urban Areas. The Cartographic Journal 46, 242–256 (2009)
17. Scacchi, W.: Process Models in Software Engineering, pp. 1–24 (October 2001)
18. Laurini, R., Servigne, S., Nol, G.: Soft real-time GIS for disaster monitoring, pp. 465–480. Springer (2005)
19. Goguen, J.A.: Reusing and interconnecting software components. Computer 19, 16–27 (1986)
20. OPT: Abstract Graphs Library Specification, Internal Report (2009)
21. Casakin, H., Barkowsky, T., Klippel, A., Freksa, C.: Schematic Maps as Wayfinding Aids. In: Habel, C., Brauer, W., Freksa, C., Wender, K.F. (eds.) Spatial Cognition II. LNCS (LNAI), vol. 1849, pp. 54–71. Springer, Heidelberg (2000)

A Flexible System for Ontology Matching*

DuyHoa Ngo, Zohra Bellahsene, and Remi Coletta

INRIA, LIRMM, Univ. Montpellier 2,
34392 Montpellier, France
firstname.name@lirmm.fr

Abstract. Most of the solutions provided by current ontology matching tools lack flexibility and extensibility namely for adding new matchers and dealing with users' requirements. In this paper, we present a system YAM++, which supports self-configuration, flexibility and extensibility in combining individual matchers. Additionally, it is more human-centered approach since it allows users to express their preference between precision and recall. A set of experiments over OAEI Benchmark datasets demonstrate its effectiveness and efficiency in terms of quality of matching and flexibility of the system.

Keywords: Ontology matching, machine learning, flexibility, self-configuration, cost-sensitive classification.

1 Introduction

Ontology matching is needed in many application domains. For example, the possibility of content-based query of the semantic Web depends only on the capacity of the system to find correspondences (mappings) between ontologies of the related information sources. Many diverse solutions of matching have been proposed so far; however, there is no integrated solution that is a clear success, which is robust, flexible for future development and usable for non expert users [19].

In this paper, we present our system YAM++, which supports self configuration, flexibility and extensibility in combining individual matchers. To demonstrate the importance of the flexibility in terms of system extensibility and user preference, let's us introduce two scenarios that frequently arise when people study schema and ontology matching.

In the first scenario, one challenge issue to researchers and developers is how to combine different similarity metrics (i.e., individual matcher [18]) in order to obtain a good performance matching system in terms of matching quality. This issue also knowns as a pre-match effort [9] in a matching process. Assume that they would like to improve the performance of the current system by adding promising similarity metrics recently published. Some issues are arisen such as (i) how to integrate the new metrics with the existing ones; and (ii) which parameters is needed to set in the new metrics. Generally, finding a good combination

* Supported by ANR DataRing ANR-08-VERSO-007-04.

and tuning these parameters are difficult and time consuming even for the domain experts. Therefore, a flexible system will be very useful to help researchers and developers to automatically integrate new metrics. To deal with the pre-match effort, we utilize machine learning techniques, which are acknowledged as a highly promising approach.

In the second scenario, imagine that a user wants to run a matching system to find all mapping pairs of entities between two ontologies. A matching system, generally, outputs a list of candidate mappings and corresponding confident values. Then, the user must verify these mappings in order to remove incorrect ones. This process will not take much time because current systems are focused on precision, and consequently the number of the incorrect mappings is small. Next, the user needs to find missing mappings which the matching system did not discover. This process is very time consuming because it will be done manually on a huge number of candidate mappings. The manual effort of this phase is called post-match effort. Users may spend many hours or even few days to finish this work. To reduce post-match effort, we make use of the cost sensitive classification approach in data mining. This approach will be explained in the section 5.3.

Based on these scenarios, the motivation of our system can be described as follows: Giving two ontologies represented in some ontology languages (N3, RDF, OWL, etc.), find a flexible approach to combine individual matchers with the following features: (i) high matching quality result (*precision, recall and f-measure*), (ii) system's self-configuration, (iii) system's extensibility, (iv) generating a dedicated matcher according to the user's preference between precision and recall.

The remainder of this paper is organized as follows: Section 2 contains the main notions used in this paper. In section 3, we describe our ontology matching system in detail. Next, Section 4 shows the prototype of our system, which will be used in experiment part. In Section 5, we present the experimental results to highlight the main interesting features of our ontology matching tool. Section 6 contains the related work. Finally, Section 7 contains concluding remarks and future work.

2 Preliminaries

In this section, we define the main notions used in this paper.

Ontology. Ontology is a formal, explicit specification of a shared conceptualization in term of concepts (i.e., classes), properties and relations [10].

Let's consider a fragments of ontologies in Fig.1. Ellipses, rounded rectangles, rectangles and dashed rectangles represent classes or datatypes, properties, instances, annotation information or data values respectively. In the schema layer arrows represent the relations between entities. Dashed arrows from data layer to schema layer indicate instances belong to classes. Arrows in the data layer indicate properties and corresponding data belong to instances.

For instance, **JournalPaper** is a class, **heading** is a property, **string** is a data type, #**a84601592** is an instance, *' A Framework for ... '* is a string value.

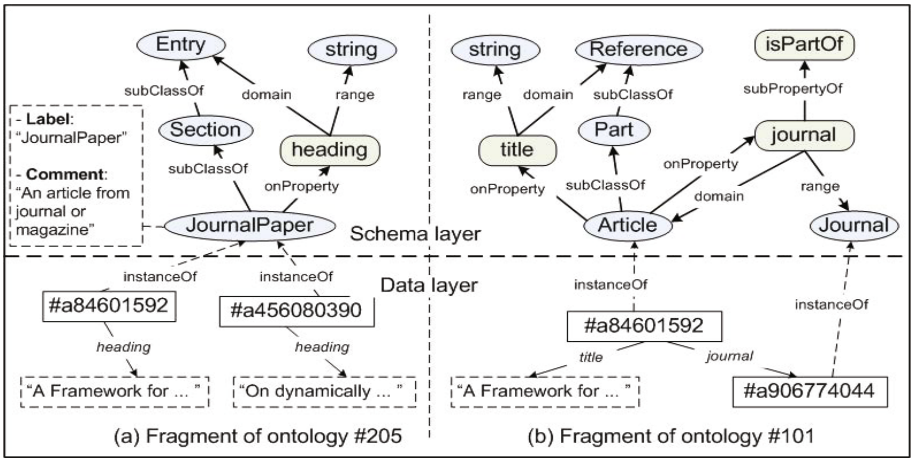


Fig. 1. Two fragments of ontologies #205 and #101 from OAEI 2010

Terminological Features. Terminological features of an entity consists of text information encoded in itself in ontology such as entities' URI, local name, labels and comments. For example, linguistic features of class **JournalPaper** include ***LocalName**: JournalPaper*, ***Label**: JournalPaper* and ***Comment**: An article from journal or magazine.*

Context Profile Features. Context profile features of an entity E consist of descriptive information of itself and its neighbors in the ontology. For example, neighbors of **JournalPaper** are class **Section**, property **heading** and instances **#a84601592**, **#a456080390**.

Similarity Metric. Similarity metric is a function of two entities that calculates their similarity value based on some extracted feature from them [8].

For example, similarity between **JournalPaper** and **Article** is 0.17 by applying ScaledLeveinstein[1] metric on their local names.

Ontology Mapping. An ontology mapping m is defined as a four-tuple of the form:

$$m = \langle e, e', r, k \rangle$$

where: e and $e'$are entities in ontologies O and O' respectively, r is a kind of relation (e.g., equivalent, subsumer) and $k \leftarrow [0,1]$ represents the degree of confidence of relation between entities [8].

Ontology Matching. Ontology matching is a process of finding all semantic correspondences between entities of different ontologies [8].

Generally, in the matching process, one or some similarity metrics are used to calculate the similarity scores between two entities. Results obtained by these similarity metrics are combined in some combination approach. The final value

[1] http://secondstring.sourceforge.net/

indicates how much entities are similar. For example, after executing the matching process we have

$$\langle JournalPaper,\ Article,\ \equiv,\ 1.0\rangle$$

It means class **JournalPaper** in ontology #205 is equivalent to class **Article** in ontology #101 with the confidence value 1.0.

3 YAM++ Ontology Matching System

3.1 YAM++ Approach

Our approach has been implemented in YAM++ - (not) Yet Another Matcher system for ontology matching. It follows the same approach used in YAM schema matching system [5]. However, the YAM++ aims to work with ontology matching, which is semantically richer than XML schema. For this purpose, we added new features such as:

– New similarity metrics working with different features (e.g., name,label, comments, relations) of ontologies' entities.
– New dictionary metrics based on different algorithms.
– New metrics based on information retrieval technique to calculate similarity score using context and descriptive information of entities.
– Graphical user interface for setting parameters, displaying and verifying discovered mappings returned from the system.

The main components of YAM++ system and the workflow are depicted in the Fig. 2

1. The inputs are at least a set of ontologies to be matched. The other additional inputs may be some "gold standard" datasets. The "gold standard" datasets here are pairs of ontologies with expert mappings between their entities provided by domain experts. They are used to generate training data.
2. Input ontologies and ontologies taken from gold standard datasets are passed to the Pre-Processing module. The ontologies are parsed and loaded into internal data structure in order to easily extract the following information:
   (a) Types of entities (class, object/data property, instance).
   (b) Descriptive information of entities such as: URI, labels, comments.
   (c) Semantic relations between entities such as: equivalent relations between entities; subsumption relation between class-class, property-property; domain and range relations between properties and classes; instanceOf relation between individuals and classes.
3. Learning data are generated by FeatureExtractor module as follows. This module exploits a variety of features of entities and calculates the similarity score between them by using similarity metrics. At this step, FeatureExtractor builds a similarity score table for two given ontologies. Each row in the

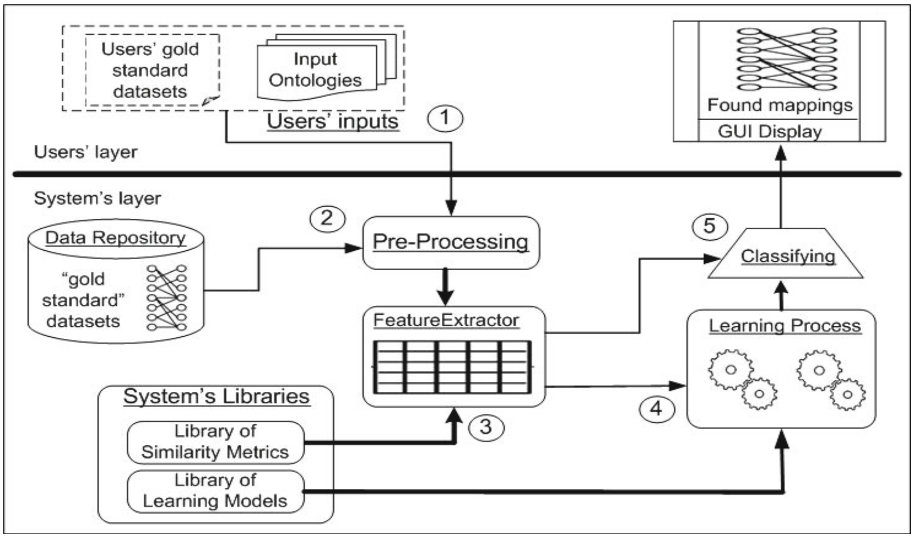


Fig. 2. Architecture and workflow

table consists of names of pair of entities (for class-class, property-property only) and an array of real values returned by applying similarity metrics on different features extracted from entities. A similarity score table build from a gold standard dataset will be converted to training data; a similarity score table build from to-be-matched ontologies will be converted to unclassified data. The rules of conversion are:

(a) Names of the similarity metrics become attribute names in machine learning instances.
(b) Each similarity score returned by a similarity metric becomes a feature value corresponding to an attribute name.
(c) The confidence values of mappings provided by expert become classes values of machine learning objects. We defined these values as feature values of attribute named **Expert/Prediction**.
(d) The only difference between training data and unclassified data is that feature values of **Expert/Prediction** attribute in unclassified data are set by unknown value.

4. Training data obtained from the previous step are passed into the Leaning Process module. In this step, all learning models are trained. They, then perform cross validation 10-folders to measure their performance. The best model with respect to the highest F-measure is selected for classifying task.
5. Each instance of unclassified data is predicted with a real value by the trained classification model obtained from the previous step. The predicted values are either 1.0 or 0.0. Instances predicted with value 1.0, are used to get the corresponding pair of matched entities. Next, discovered mappings are displayed in the graphical user interface.

6. In GUI (Graphical User Interface), ontologies are displayed as hierarchies of entities (only classes and properties). Each mapping discovered by the system is represented as a line connecting two entities of two ontologies. The user can select and validate mappings or discover new mappings manually.

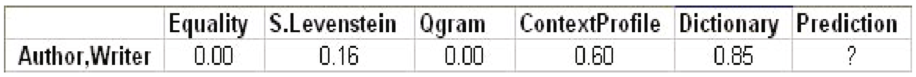
3.2 An Example of Learning Model

Now, let us illustrate an example. Assume that YAM++ system measures the similarity between entities by using 5 similarity metrics as follows:

- Equality method returns 1.0 if the labels of two entities are identical, otherwise it returns 0.0.
- ScaledLevenstein and QGram are welknown edit-based and token-based string metrics respectively. Refer to SecondString [2] library for more detail.
- ContextProfile metric measure the similarity score between two entities by computing cosine similarity of their text profiles. See [16] for more detail about creating context profiles for each entity.
- Dictionary metric measures the similarity score between entities by comparing the meaning of their name in a dictionary (e.g WordNet).

Without loss of generality, YAM++ combines these metrics by using a decision tree model. After training process, a tree model is configured as Fig. 3. Here, a decision tree is a tree whose non-leaf nodes are the similarity metrics, leaf nodes values are either 1.0 ore 0.0 indicating if there is a match or not. At a non-leaf node, a similarity value of to-be-mached entities is computed by the similarity metric in ongoing node. The returned value is compared with condition values on outgoing edges from current node in order to decide which child node will be reached. This process will start at root node and iterate until a leaf node is reached. The value of destination leaf node indicates whether the two entities should match or not.

Suppose that we need to match a pair of entities (**author**, **writer**). The feature vector obtained by running 5 similarity metrics is follows:



	Equality	S.Levenstein	Qgram	ContextProfile	Dictionary	Prediction
Author,Writer	0.00	0.16	0.00	0.60	0.85	?

Here, the question symbol in Prediction field means that the similarity of two entities is unknown. According to the trained tree, the decision path is: Equality -> SclaledLevenstein -> Dictionary -> 1.0 . It means two entities are matched.

Evidently, each path from the root node to each leaf on the tree is a combination of subset similarity metrics. In this example, only 3 among 5 metrics are combined to produce the final result.

[2] http://secondstring.sourceforge.net/

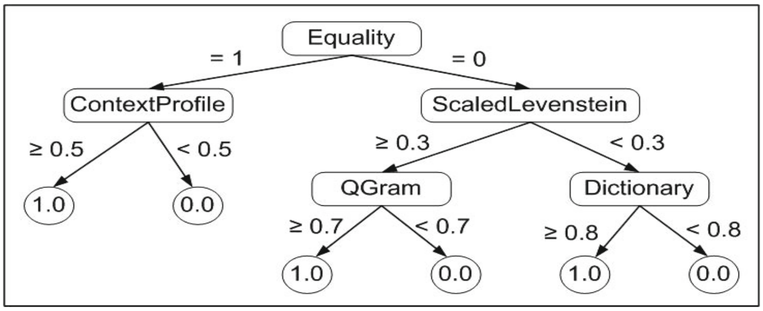


Fig. 3. Combining similarity metrics with a decision tree model

4 System Prototype

4.1 Feature Extraction and Similarity Metrics

Due to the heterogeneity of ontologies, there is no single metric which could be sufficient to measure the similarity of entities in all matching cases. Therefore, even for each extracted feature from entities, we should use different similarity metrics with the hope that they will complement to each other.

YAM++ supports more than 30 different similarity metrics. These metrics are divided into three main groups: (i) **name metrics** - which compare entities by their URI names; (ii) **label metrics** - which compare entities by their labels; and (iii) **context metrics** - which compare entities by their descriptive and context information.

Name and label metrics belong to Terminological category [8]. Most of String-based metrics (Levenstein, SmithWaterman, Jaro, JaroWinkler, MongeEklan,etc.) are taken from open source libraries SecondString [3] and SimMetrics [4]. We also have implemented popular metrics such as Prefix, Suffix, LCS [8], SubString - Stoilos [20]. In term of Language-based metrics, we developed three well-known metrics corresponding to Lin, JingConrath and WuPalmer algorithms [14]. These metrics use Wordnet [5] as an auxiliary local thesaurus. Additionally, we proposed a hybrid approach to combine string-based and language-based metrics. The detail of algorithms can be found in our paper [16].

Context metrics used in YAM++ belong to the both Structural and Extensional categories [8]. The main idea behind these metrics is described as follows. We build three types of context profile (i.e. IndividualProfile, SemanticProfile and ExternalProfile) for each entity. Whereas IndividualProfile describes annotation information of individual entity, SemanticProfile and ExternalProfile describe the information of the relationships between entities on ontology

[3] http://secondstring.sourceforge.net/
[4] http://sourceforge.net/projects/simmetrics
[5] http://wordnet.princeton.edu

(i.e., entity-entity and entity-instances). These context profiles then can be used to compare entities by some information retrieval techniques. See [16] for more detail.

4.2 Attributes and Training Data

Theoretically, all metrics can be used as selected attributes, but it will make the learning and classifying processes very time consuming. Therefore, we propose a strategy to select the most suitable similarity metrics for each of three groups (i.e., Name metrics, Labels metrics and Context Profile metrics). The heuristic of selection is based on hypothesis "A good feature subset is one that contains features highly correlated with the class" [11].

The selection of attributes works as follows. The correlation value is calculated by Pearson's formula[6] between similarity scores obtained by a metric and values provided by experts for each test in **Benchmark**[7] datasets. Finally, the similarity metrics having the highest average values in each of three groups above are selected. Table 1 shows a set of candidate similarity metrics used for the experiments section.

Table 1. List of selected similarity metrics

Similarity Metrics	Descriptions
Levenshtein, SmithWaterman, Stoilos	String-based metrics which compare entities by their local names. String pre-processing is not needed.
QGramDistance	String-based metric which compares entities by their local names. String is split into tokens with length 3.
MongeEklanStoilos	String-based metric which compare entities by their local names. Pre-processing procedure splits string into tokens. Tokens are compared by Stoilois metric.
TokLinStoilois, TokWuPalmerStoilois	Hybrid metrics which compare entities by local names. Pre-processing procedure splits string into tokens. Tokens are compared by combination of string-based (Stoilois) and linguistic-based metrics (Lin and WuPalmer).
LabelLinStoilois, LabelWuPalmer-Stoilois, LabelIndentity	Hybrid metrics which compare entities by their labels. Labels are compared by LinStoilois, WuPalmerStoilois and synonym methods after running pre-processing procedure. The maximum similarity score between labels will be selected.
FullContextProfile	Information Retrieval metric compare entities by their context profile. A maximum similarity score computed by IndividualProfile, SemanticProfile and ExternalProfile will be selected.

In order to provide training data for learning models, we use **Benchmark** datasets. Tests in Benchmark datasets play the role of "gold standard" in our

[6] http://en.wikipedia.org/wiki/Correlation_and_dependence

[7] http://oaei.ontologymatching.org/2010/benchmarks

approach. However, we do not select all tests to generate training data in order to reduce noise data. For example, according to the expert alignment file, class **Chapter** in ontology #101 matches with class **dzqndbzq** in ontology #202 but all of our metrics compute similarity score for this pair with very low values (for instance, Levenshtein("Chapter","dzqndbzq")=0.0). This mapping will produce a noise in training data and it will degrade the performance quality of learning model. To solve this problem, we propose a strategy to select suitable gold standard datasets for a set of selected similarity metrics. It is based on the following heuristic: a training data is representative with respect to a feature if this feature is highly correlated to the class.

The selection of **candidate "gold standard" datasets** for YAM++ works as follows. Firstly, the system builds a correlation table for a set of selected similarity metrics with all tests from Benchmark datasets. For each test, the system computes the average correlation coefficient of all metrics. Next, only tests having the average of correlation values higher than threshold are selected.

4.3 Learning Models

YAM++ can work with different machine learning algorithms in order to build a classification model. These algorithms are divided into 5 groups as follows:

– Tree-based: J48, J48Graft, ADTree, SimpleCart, NBTree.
– Probability-based: NaiveBayes, BayesNet.
– Function-based: Logistic, MultiLayerPerceptron.
– Rule-based: JRip, VFI, DecisionTable.
– Instance-based: IBk, NNGe.

The implementation of these algorithms are taken from open source library Weka[8]. To get more information about these algorithms, see [22].

4.4 Evaluation Criteria

We follow the evaluation criteria used in the OAEI campaign. The harmonic mean of precision, recall and f-measure are used to evaluate the performance quality of a matching system over a set of tests. Assume that we have n tests. Let i indicates ith test; $|R_i|$ refers to the number of reference mappings provided by expert domain, $|A_i|$ is the total number of mappings discovered by a matching system and $|C_i|$ is the number of correct mappings.

– Harmonic mean of *precision*, $H(p) = \frac{(\sum_{i=1}^{n} |C_i|)}{(\sum_{i=1}^{n} |A_i|)}$;
– Harmonic mean of *recall*, $H(r) = \frac{(\sum_{i=1}^{n} |C_i|)}{(\sum_{i=1}^{n} |R_i|)}$;
– Harmonic mean of *f-measure*, $H(fm) = \frac{2 * Hp * Hr}{(Hp + Hr)}$;

[8] http://www.cs.waikato.ac.nz/ml/weka

5 Experiments

In this section, we present the capabilities of our system using three experiments:

1. We compare different machine learning models and select the best one.
2. We show the flexibility and extensibility of our system in terms of integrating new similarity metrics automatically and transparently to the user.
3. We show the ability to generate a dedicated matcher based on the user's preference (promoting recall).

To evaluate the matching quality, we perform experiments on Benchmark datasets which are published on OAEI 2010 campaign [9]. We adopt these tests because they are widely used in evaluation of many ontology matching systems. All experiments are executed with JRE 6.0 on Intel 3.0 Pentium, 3Gb of RAM, Window XP SP3.

5.1 Experiment 1: Comparison of Performance of Learning Models

The aim of this experiment is to find the most suitable learning model for our approach. To do that, we randomly select a set of "gold standard" datasets to generate training data and then measure the performance of each learning model by applying 10-fold cross-validation technique [22]. This process is repeated 30 times in order to limit the impact of randomness during the evaluation. The average of F-measure values of all learning models are displayed in Fig.4. The model which has the highest performance is J48 - a modified version of the decision tree model. Following the J48 model are J48graft, JRip and SimpleCart models. According to the comparison result, hereafter, we use J48 model in the learning and classification tasks.

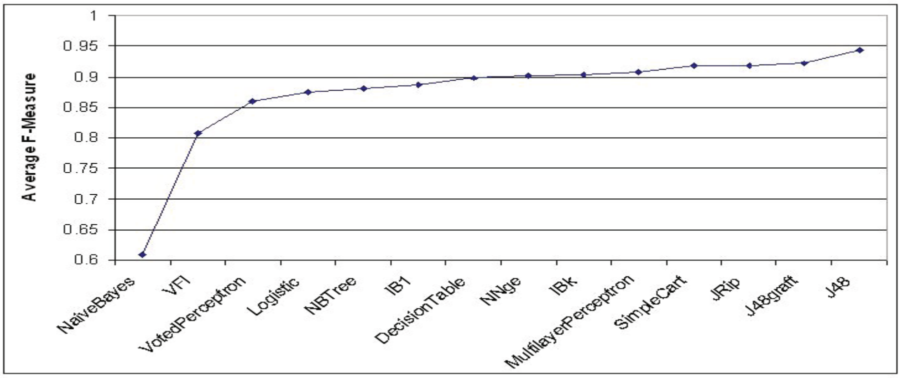


Fig. 4. Comparison of the performance of learning models

[9] http://oaei.ontologymatching.org/2010/benchmarks/

5.2 Experiment 2: Flexibility, Extensibility and Self-Configuration

According to [19], matcher combination and self-tuning is a big challenge to all matching systems. Finding the most appropriate parameters, such as thresholds, weights, and coefficients is very difficult and time consuming even for experts. In this experiment, thanks to the advantages of machine learning technique, we show that YAM++ can deal with that challenge.

For demonstration purpose, we show the abilities of YAM++ such as flexibility, extensibility and self-configuration in this experiment. The main idea is that we step by step add new metrics to the system and then see the changes of the system's performance. The whole process runs automatically without any manual tuning parameters. The setting of the experiment is described as follows:

– According to the result of previous experiment, we use J48 as a classification model.
– Training data is generated from randomly selected set of the candidate "gold standard" datasets. The training data remains the same for one execution.
– To measure the performance of YAM++, we use all tests in Benchmark datasets. The Benchmark datasets include 111 tests. Each test consists of source (reference) ontology and a test ontology, which is created by altering some information from the reference.
– For each time of execution, YAM++ runs with three scenarios corresponding to three groups of similarity metrics. Initially, the first group consists of all of the name metrics. Then, label metrics are added to become second group. Finally, in the third group, context metrics are supplemented.

We repeat this process 30 times to limit the impact of the randomness in learning and classifying processes. The Fig.5 displays the result of the experiment.

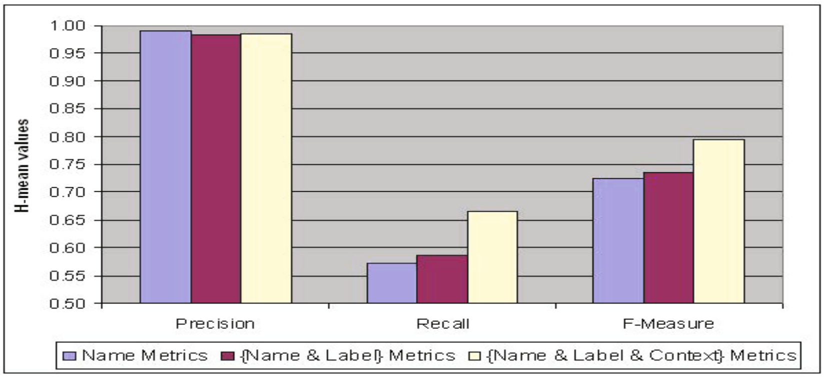


Fig. 5. Performance of YAM++ with the different sets of similarity metrics

The observations from the experiment are:

- Generally, YAM++ achieves very high **precision** (≈ 1.0) with the three groups of similarity metrics. It means that the selected similarity metrics by our proposed strategy obtain high accuracy in calculating similarity scores between entities. Besides, the **recall** and **f-measure** values are increased when adding new metrics. It is in accord with the general trend: the more features are exploited and more appropriate metrics are used, the more mappings will be discovered.
- Let see an example with the group of tests **#201, #201-2, #201-4, #201-6 and #201-8**. Names and labels of entities in these tests are replaced by random strings with proportion **100%**, **20%**, **40%**, **60%** and **80%** respectively. However, they support annotation information and data instances for entities.
  - By using only name metrics, YAM++ obtains the average **recall** equal **0.41** in this group. Although this value is low but it is the expected number. Let see the description on test ontologies in this group. Names of entities are replaced by random strings with proportion 20%, 40%, 60%, 80% and 100%. The average number of altered names of entites is $(20\% + 40\% + 60\% + 80\% + 100\%)/5 = 60\%$. It mean that the maximum number of mappings found by name metrics is $100\% - 60\% = 40\%$. This number is in line with the recall value (**41%**) obtained by our system.
  - By adding metrics exploiting label feature, the recall increases to **0.57**. It is not much improvement ($0.57 - 0.41 = 16\%$) because only small number of entities in test ontologies support labels as complement to their names. For example, entity **dzqndbzq** in test ontology **#201** has label "BookPart".
  - Next, by adding label and context metrics, the recall increase to **0.96**. It is a significant improvement ($0.96 - 0.41 = 55\%$). This improvement is due to the fact that all entities in test ontologies support descriptive information. Additionally, data instances are provided for most concepts in ontologies.
- Finally, the most interesting feature to note is that the process of reconfiguration system with new metrics is totally automatic and transparent to the users. For each group of similarity metrics, YAM++ will find the best configuration to combine these metrics without any parameter tuning by hand. An example of combination similarity metrics by J48 (decision tree model) can be seen at section 3.2

5.3 Experiment 3: Promoting Recall

Traditionally, the measure used to compute performance quality of matching tools, is the F-Measure: a combination of Precision (the ratio of correctly found correspondences (a.k.a true positive) over the total number of returned correspondences [8]) and Recall (the ratio of correctly found correspondences over the total number of expected correspondences [8]), in which precision and recall have the same weight. F-Measure makes sense when using a matching tool as black box, without any user validation. But, most of the time, the user have to perform some post-match effort in order to discard some irrelevant and discover the missing mappings. In this experiment, we demonstrate the impact of user preference between Precision and Recall on post-match effort.

Technically, most classification models suffer from two errors during classifying: i) discovering an irrelevant correspondence (a.k.a. false positive) and ii) missing a relevant correspondence (a.k.a. false negative). The first error decreases precision while the second one decreases recall. In order to get better result in term of recall, we need to set the cost on false negative error higher than that on false positive error. This is a well-known issue called Cost-Sensitive Learning in Data Mining [7].

In order to deal with cost-sensitive learning, we use MetaCost and CostSensitiveClassification algorithms [22]. These algorithms belong to meta-learner class. They make a wrapper on base learning models in such a way that learning models effectively minimize the cost. The preference between Precision and Recall is expressed by a proportion of the cost on false negative and the cost on false positive.

The setting of this experiment is described as follows:

- We use all similarity metrics from name, label and context groups.
- To see the effect of promoting recall, we perform matching with YAM++ on the group **3xx** {**#301, #302, #303, #304**}. They are known as real test ontologies in the bibliographic domain.
- Training data is generated in the same way used in previous experiment. However, we remove all tests from group **3xx** out of the selection of "gold standard" datasets.
- We use J48 model as a base learner. In each execution time, we set proportion between the cost on false negative and the cost on false positive in a MetaCost model to have a cost sensitive learning model. The cost ratio values are selected from 1 to 15. We use these values because they were considered in other studies [3,4,1] as providing good results.

We repeat this process 10 times and for each time, a new training data is generated. Results of matching YAM++ with group **3xx** are shown in the Fig.6.

The observations from these figures are:

- In the top figure, the general trend is increasing for both number of True Positive and False Positive discovered mappings, whereas the number of False Negative decreases. The trend is reasonable with respect to cost sensitive learning. In fact, the goal of cost sensitive classification is to find the frontiers between class regions (i.e, "Correct " and "Incorrect" mapping in our case), explicitly or implicitly. When we set the cost on false negative higher than the cost on false positive, we aim to reduce the number of misclassifying correct matches as incorrect ones. Thanks to algorithm of minimizing expected cost [3], the frontier of discovered mapping will be enlarged and more correct mappings will be discovered.

– In more detail, when the cost ratio values are set from 1 to 8, both the numbers of True Positive and False Positive increase. At the point where the cost is set to 8, YAM++ discovers **21** (175-154) additional True Positive, but **34** (56-22) additional false positives in comparison with not using cost sensitive technique.
– Let see the bottom figure, obviously, Recall goes up until the cost is equal to 9, then stays around value **0.76**. On contrary, Precision goes down and then becomes stability around **0.78** since the cost is set to 10. The trend of F-Measure is not significantly changed. It stays around **0.77** during the process of using cost sensitive learning.
– Notice that whatever the value we set for proportion, it always remains some matches we are not able to discover automatically. The post-match process is always needed to handle with missing or irrelevant/incorrect mappings.
– In fact, the effort for manually removing an incorrect mapping is much less than the one for discovering a new correct mapping among nearly **10000** pairs (total candidate mappings of 4 tests in group **3xx**). Therefore, by promoting recall, our system reduces the user's post-match effort during the validation phase.

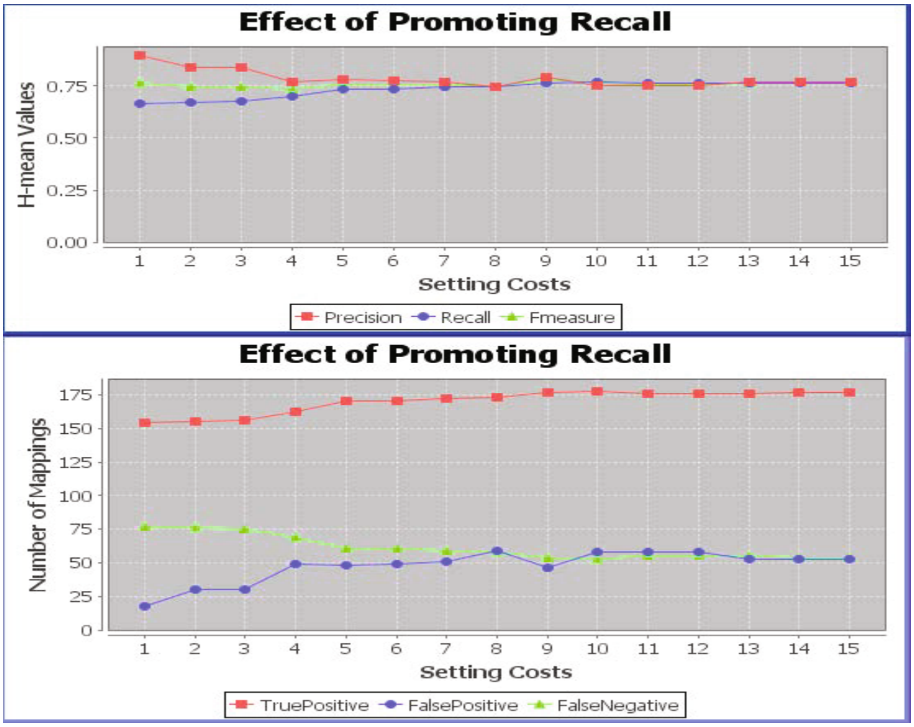


Fig. 6. The impact of promoting recall on the number of discovered mappings

6 Related Work

There are many studies on Ontology Matching [13,8]. In this section, we only mention the closest ones that are based on machine learning approaches.

GLUE [2] is a well-known of learning-based ontology mapping system. GLUE uses a set of base learners to exploit different type of information from instances and taxonomy structures. Then, it uses a meta-learner to combine these base learners to achieve higher classification accuracy than any single base learner alone. The drawback of GLUE is that it requires a large number of instances associated with the nodes in taxonomies, whereas most ontologies do not contain these information. YAM++ is different with GLUE in that YAM++ uses machine learning approach to combine simple similarity measures which exploit different features of entities such as name, description and structure information.

In [6], the proposed approach makes use of meta level learning techniques to improve the matching performance. It does not work directly with simple similarity measures like YAM++ does, but instead, it works with existing ontology matching systems. The result of learning process is to produce a best combination of those existing matchers. Similarly, in [17], the proposed approach evaluates the confidence value of mappings for ontology alignments using the rules extracted through machine learning methods. The basic idea is to convert the extracted rules into formulas that reflect the reliance of each rule. Rules are further combined in order to generate a single value on the mapping.

Our approach is in line with approaches described in [15,12,21] in the way of using machine learning approach to combine different similarity metrics. However, the additional contribution in YAM++ is that we applied data mining techniques to help users to reduce the post-match effort.

7 Conclusion

In this paper, we present a flexible system for ontology matching task that proves the following interesting features:

– Flexibility and extensibility in terms of combining different similarity measures.
– Generating a dedicated matcher according to the user's preference.

We have developed a prototype which has been tested with the datasets of OAEI 2010 benchmark. Through these experiments, we have validated the features listed above. However, most of the similarity metrics used in the current system are terminological and contextual metrics. In the future work, we plan to integrate structural and semantic methods to our system in order to improve its performance.

References

1. Chawla, N.V., Cieslak, D.A., Hall, L.O., Joshi, A.: Automatically countering imbalance and its empirical relationship to cost. Data Min. Knowl. Discov. 17(2), 225–252 (2008)

2. Doan, A., Madhavan, J., Domingos, P., Halevy, A.Y.: Ontology matching: A machine learning approach. In: Handbook on Ontologies, pp. 385–404 (2004)
3. Domingos, P.: Metacost: A general method for making classifiers cost-sensitive. In: Knowledge Discovery and Data Mining, pp. 155–164. ACM Press (1999)
4. Drummond, C., Holte, R.C.: Cost curves: an improved method for visualizing classifier performance. In: Machine Learning, pp. 95–130 (2006)
5. Duchateau, F., Coletta, R., Bellahsene, Z., Miller, R.J.: Yam: a schema matcher factory. In: CIKM Conference, pp. 2079–2080 (2009)
6. Eckert, K., Meilicke, C., Stuckenschmidt, H.: Improving Ontology Matching Using Meta-level Learning. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 158–172. Springer, Heidelberg (2009)
7. Elkan, C.: The foundations of cost-sensitive learning. In: Artificial Intelligence, pp. 973–978 (2001)
8. Euzenat, J., Shvaiko, P.: Ontology matching. Springer, Heidelberg (2007)
9. Dillon, T.S., Chang, E., Meersman, R., Sycara, K.: Advances in Ontology Matching. In: Dillon, T.S., Chang, E., Meersman, R., Sycara, K. (eds.) Advances in Web Semantics I. LNCS, vol. 4891, pp. 1–6. Springer, Heidelberg (2008)
10. Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acquisition 5, 199–220 (1993)
11. Hall, M.A.: Correlation-based feature selection for discrete and numeric class machine learning. In: ICML, pp. 359–366 (2000)
12. Ichise, R.: Machine learning approach for ontology mapping using multiple concept similarity measures. In: ICIS Conference, pp. 340–346. IEEE Computer Society, Washington, DC (2008)
13. Kalfoglou, Y., Marco Schorlemmer, W.: Ontology mapping: The state of the art. In: Semantic Interoperability and Integration (2005)
14. Lin, F., Sandkuhl, K.: A Survey of Exploiting Wordnet in Ontology Matching. In: Bramer, M. (ed.) Artificial Intelligence and Practice II. IFIP AICT, vol. 276, pp. 341–350. Springer, Heidelberg (2008)
15. Mao, M., Peng, Y., Spring, M.: Ontology mapping: As a binary classification problem. In: Semantics, Knowledge and Grid Conference, pp. 20–25 (2008)
16. Ngo, D., Bellasene, Z., Coletta, R.: A generic approach for combining linguistic and context profile metrics in ontology matching. In: ODBASE Conference (2011)
17. Maio, N.S.P., Bettencourt, N., Rocha, J.: Evaluating a confidence value for ontology alignment. In: Ontology Matching Workshop (OM 2007) (November 2007)
18. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. 10(4), 334–350 (2001)
19. Shvaiko, P., Euzenat, J.: Ten Challenges for Ontology Matching. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part II. LNCS, vol. 5332, pp. 1164–1182. Springer, Heidelberg (2008)
20. Stoilos, G., Stamou, G., Kollias, S.: A String Metric for Ontology Alignment. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 624–637. Springer, Heidelberg (2005)
21. Svátek, V.: Combining ontology mapping methods using bayesian networks. In: ISWC Workshop (2006)
22. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (October 1999)

A CASE Tool to Support Automated Modelling and Analysis of Security Requirements, Based on Secure Tropos

Michalis Pavlidis, Shareeful Islam, and Haralambos Mouratidis

School of Architecture, Computing, and Engineering, University of East London, UK
m.pavlidis@ieee.org, {shareeful,haris}@uel.ac.uk

Abstract. Secure Tropos, an extension of the Tropos methodology, considers security requirements alongside functional requirements, from the early stages of the system development process. The Secure Tropos language uses security concepts such as security constraint, secure goal, secure plan, secure resource, and threat to capture the security concepts from both social and organisational settings. These concepts are used to model and reason about security for a specific system context. This paper presents a CASE tool, called SecTro, which supports automated modelling and analysis of security requirements based on Secure Tropos. The tool's architecture, layout, and functionalities are demonstrated through a real world example using the Secure Tropos concepts.

Keywords: Security, Goal Modelling, Requirements Engineering, Secure Tropos, SecTro, and CASE tools.

1 Introduction

Information systems play an important role in education, health care, banking, and transportation. Ensuring security of information systems is a vital and challenging task. Such systems often store sensitive customer and business information, which needs adequate protection [1]. There should be cost effective and realistic protection mechanisms to protect such systems against any potential attack. It is already agreed, by the industry and relevant research communities, that security should be considered from the early phases of the development process [2]. Identifying and analysing the security requirements along with the functional requirements provides a better comprehension of the system's security issues and limits the conflicts between the security and functional requirements for more secure information systems [3, 4, 5].

Secure Tropos is a security requirements engineering methodology that considers security issues throughout the whole development process [6]. The approach identifies, models and analyses the security concepts within the organization and social settings from the early stages of development. However, considering security issues from the early development stage may increase the number of activities during the development, which may not be always affordable depending on the project's specific constraints. Therefore, there is the need for an automated tool to support security modelling activities based on the Secure Tropos concepts [3]. This paper demonstrates a tool, named SecTro, which supports the designers during the security modelling activities and assists them in producing the output based on the activities. The tool demonstration includes the architecture, the layout, and the functionalities based on a real example.

The rest of the paper is structured as follows. Section 2 is a review of Secure Tropos. Section 3 illustrates the tool that supports Secure Tropos. Section 4 discusses the related work, while section 5 concludes the paper and presents future work.

2 Secure Tropos Methodology

Secure Tropos is an extension of Tropos methodology that takes security into account. Secure Tropos considers the basic Tropos concepts such as dependency, goal, task, resource, and capability and adds security concepts such as security constraint, secure goal, secure plan, secure resource, and secure capability [6, 7]. Secure Tropos includes the following modelling activities:

- Security reference modelling. The security reference modelling activity involves concepts such as the security features of the system under development, the protection objectives of the system that contribute positively towards the security features, the security mechanisms that contribute positively towards the protection objectives, and the threats that have a negative contribution towards the security features of the system. The security reference diagram can be used as a reference point during the development process.
- Security constraint modelling. Involves the modelling of the security constraints imposed to the actors and the system, and it allows developers to perform an analysis by introducing relationships between the security constraints or a security constraint and its context.
- Secure entities modelling. Involves the analysis of the secure entities of the system and it is considered complimentary to the security constraints modelling.
- Secure capability modelling. Involves the identification of the secure capabilities of the actors that guarantee the satisfaction of the security constraints.

In addition, Secure Tropos consists of four stages:

- Early requirements. Stakeholders are modelled as social actors, which depend on each other in order to achieve goals, carry out plans or deliver resources. The outcome of this phase is an actor diagram and a number of goal diagrams.
- Late requirements. In the late requirements the actor diagram of the early requirements is extended with the introduction of the system to be as an actor that has a number of dependencies with the rest of the social actors. In fact, these dependencies will be the functional and non-functional requirements of the system.

- Architectural design. According to the system requirements the designer in this phase defines the global architecture of the system, which consists of a number of sub-systems. Sub-systems are interconnected with data and control flows that are represented as dependencies. Then, each actor is assigned to an agent who has a number of capabilities. Usually, during the end of the architectural design the security attack testing takes places, where the design of the system is tested against the security requirements.
- Detailed design. In detailed design there is further specification of the agent capabilities and the interactions between the agents. Agent capabilities and their included plans are modelled with the help of UMLsec activity diagrams, while agent interactions are modelled with the help of UMLsec sequence diagrams.

The meta-model of Secure Tropos [8] is shown in Fig. 1 and more detailed information about Secure Tropos can be obtained from [6, 7].

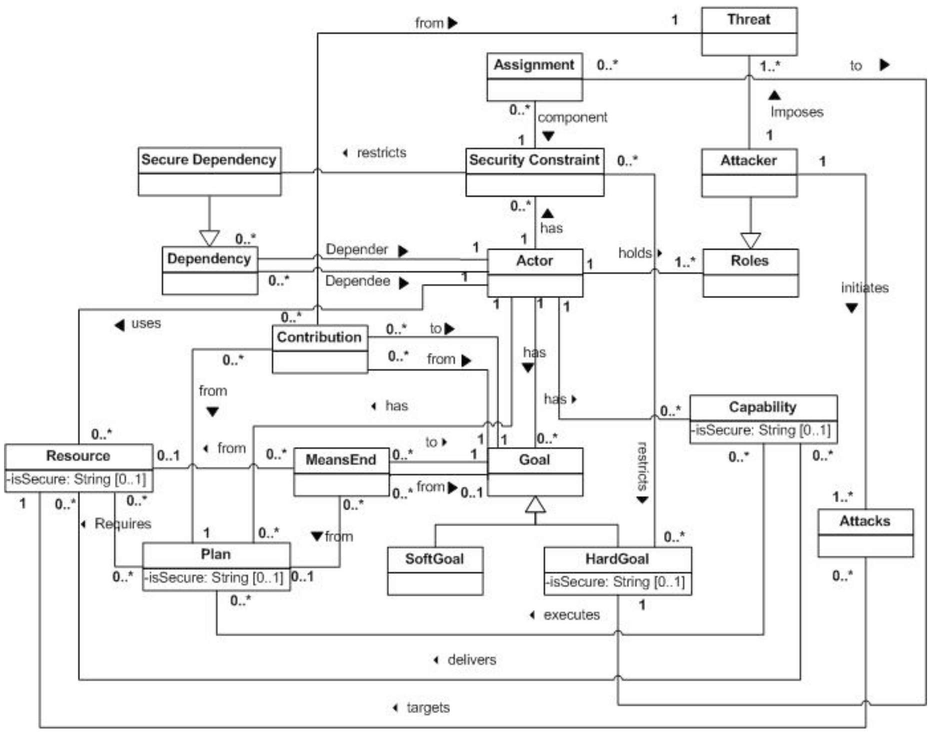


Fig. 1. Secure Tropos meta-model

3 The SecTro Tool

3.1 SecTro Architecture

The SecTro tool [9, 10] was developed to support the Secure Tropos modeling activities for the creation of the visual models and to assist the designers with the automation of some aspects of the methodology. It also includes a basic validation of the developed models. It is a standalone application that was built with the Java programming language, which makes the tool usable across different platforms.

The package diagram of SecTro is shown in Fig. 2 and a description of the packages is given in Table 1. The main classes for the drawing functionalities of the tool are shown by a class diagram in Fig. 3. The class diagram of the graphical user interface (GUI) package is shown in Fig. 4. All the elements that can be drawn, such as an actor and a hard goal, belong to the ElementType class, while all the links between the elements, such as "plays" and "satisfies" links, belong to the LinkType class.

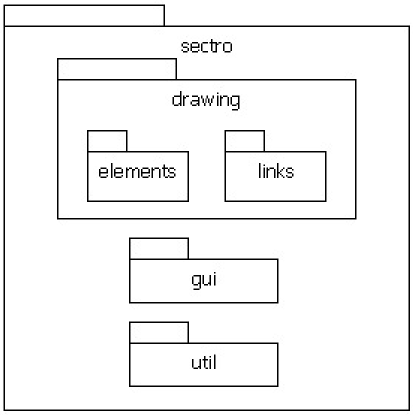


Fig. 2. Package diagram of SecTro

Table 1. Description of the SecTro packages

Package	**Description**
sectro	The parent package that includes the main class and all the sub packages
sectro.drawing	Contains the generalized class for all the drawing objects (DrawingObject) and the elements and links packages
sectro.drawing.elements	Contains the classes for all the drawing elements (Actor, HardGoal,Resource, Plan, etc.)
sectro.drawing.links	Contains the generalized class for all the Links (Link) and the classes for all the drawing links (LinkDependency, LinkRestricts, LinkPlays, etc.)
sectro.gui	Contains all the classes related to the user interface (MainForm, ToolBar, MenuBar, etc.)
sectro.util	Contains all the utility classes (ImageUtil, XMLUtil, FileUtil, etc.)

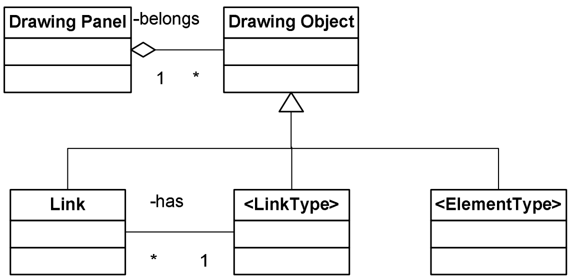


Fig. 3. Class diagram of the SecTro drawing functionality

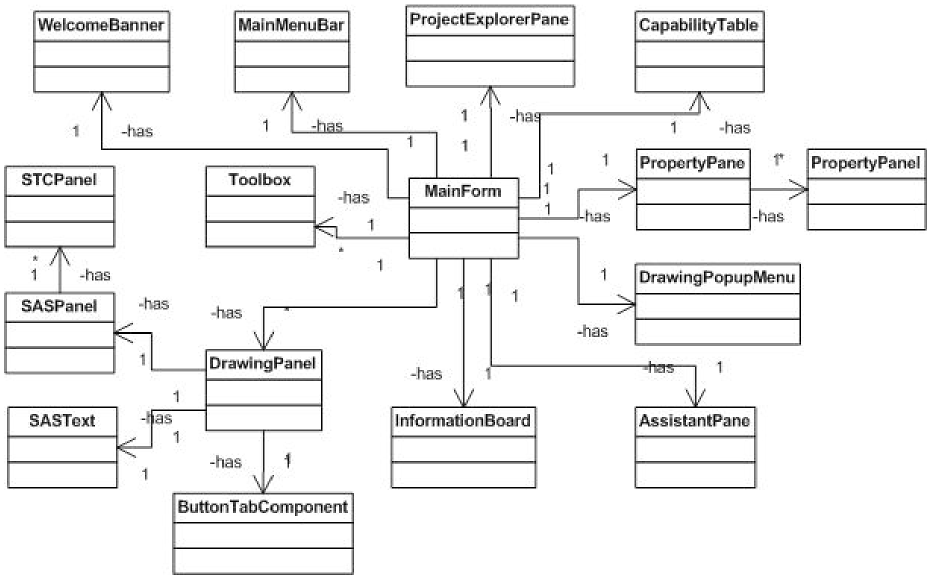
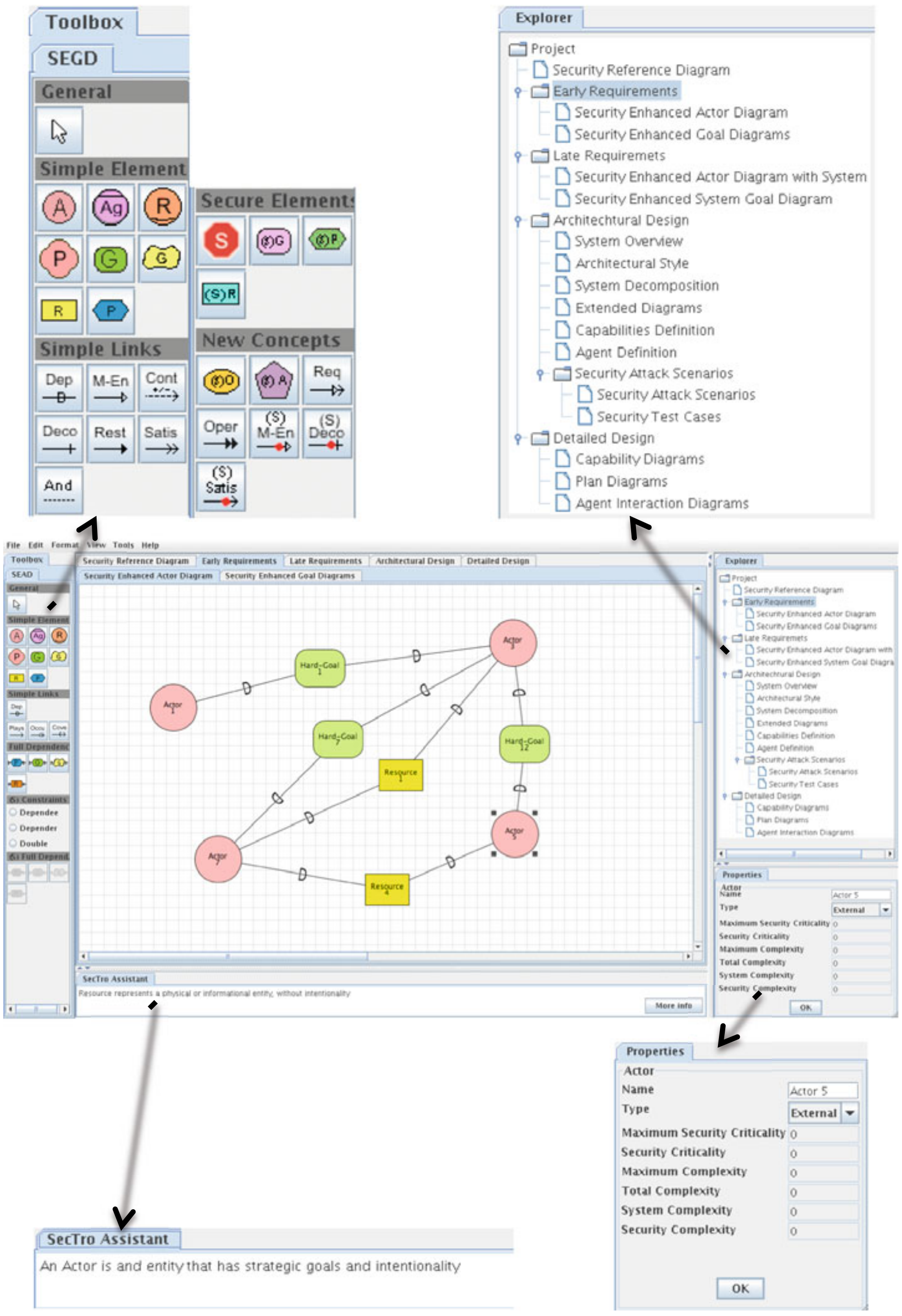


Fig. 4. SecTro GUI class diagram

3.2 SecTro Layout

SecTro's workspace, as shown in Fig. 5, consists of the drawing canvas in the centre of the workspace. The drawing canvas is the space where the designer is drawing the models. The drawing area generally automatically expands depending on the developing model size. On the top of the workspace, there is a series of tabs for showing the developed diagrams for each stage of Secure Tropos. The project explorer and the properties panel are on the right side and the toolbox is on the left side of the workspace. Apart from the tabs, the project explorer is another way for the designer to navigate through the models. Below the project explorer is the properties panel that shows information about the properties of a selected element of a model.



SecTro Assistant

Resource represents a physical or informational entity, without intentionality

SecTro Assistant

An Actor is and entity that has strategic goals and intentionality

Properties

Actor

Name	Actor 5
Type	External
Maximum Security Criticality	0
Security Criticality	0
Maximum Complexity	0
Total Complexity	0
System Complexity	0
Security Complexity	0

OK

Fig. 5. SecTro workspace

The toolbox on the left contains the graphical elements of the Secure Tropos that can be added to a model. However, not all elements appear on the toolbox, but only the ones that can be used at a specific stage and model. The reason for that is to ensure the syntactical correctness of the developing models by preventing the designer to add invalid elements. Therefore, the elements of the toolbox change according to the tab that is selected. At the bottom of the workspace is the SecTro Assistant that shows information and feedback to the designer about his actions. The graphical representations of the concepts of Secure Tropos, by the SecTro tool are shown in Fig. 6.

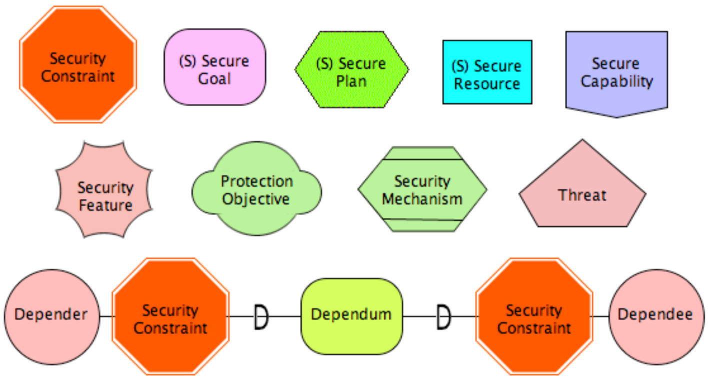


Fig. 6. Secure Tropos notation

3.3 SecTro Functionalities Supporting the Security Aware Process

The main functionalities of the SecTro are to support the security modelling activities of Secure Tropos. Therefore, the tool enables the designer to perform security reference modelling, security constraint modelling, secure entities modelling, and secure capability modelling activities.

To demonstrate the tool functionalities, we use in the rest of the section a real world example. The example is about a value added application for UK located bank that offers its customers use of smart cards (debit/credit card) to pay for purchases. To support that functionality; the bank collaborates with some retailers and the card issuer for the smart card based payment infrastructure. Security is an important aspect for this application because customer identifiable and sensitive data is handled by the participating parties. Based on the above, we identify the following important concepts.

Goals: protect personal data, secure processing.
Actors: customer, application providers (bank, card issuer) and retailer.
Plans: transaction and purchase detail, and update account balance, request payment from the customer account.
Resources: customer identifiable data, customer accounts information.

Security Constraints: Only legitimate customer, transfer minimum data.
Attacker goal: obtain customer data, unauthorised access.

During the early requirements stage the designer constructs the security reference diagram, the actor diagram and a number of goals diagrams. Figures 7, 8, and 9 present screenshots of the aforementioned diagrams created for the purpose of the paper. The designer has the option to construct the goal diagrams in the initial actor diagram or create a new tab in order to construct a goal diagram separately from the actor diagram and the rest of the goal diagrams.

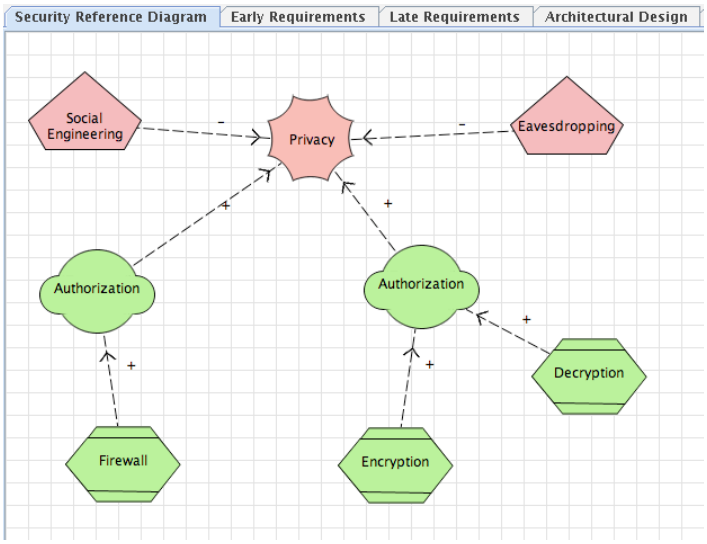


Fig. 7. Security reference diagram

In the late requirements stage the designer constructs the actor diagram with the system and the goal diagram of the system. Similarly, the system's goal diagram can be constructed in the actor diagram with the system tab or the designer can choose to construct it in the dedicated tab. Figures 10 and 11 present an actor diagram with the system and a goal diagram of the system created for the purpose of this paper.

In the actor diagram with the system the dependencies of the system with the rest of the actors are the functional requirements of the system. In the system goal diagram the designer specifies what the system needs to do in order to satisfy the requirements. The security constraints in the system goal diagram are the security requirements of the system. The designer specifies how the system satisfies these security requirements through the concepts of secure goal, plan and resource. This process is not sequential and the designer can goal back at the actor diagram with the system and add more security constraints. The system goal diagram will be updated automatically in order to include the new security constraint.

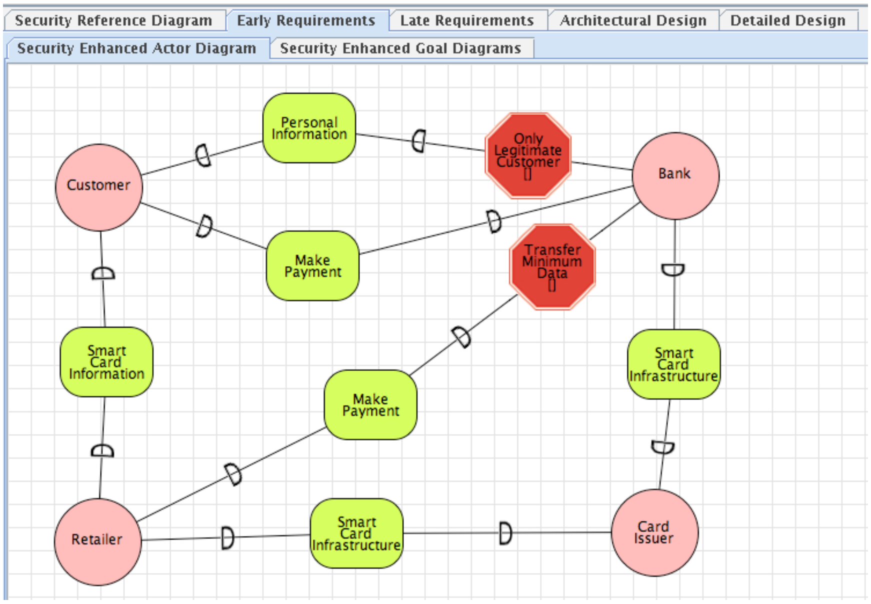


Fig. 8. Actor diagram

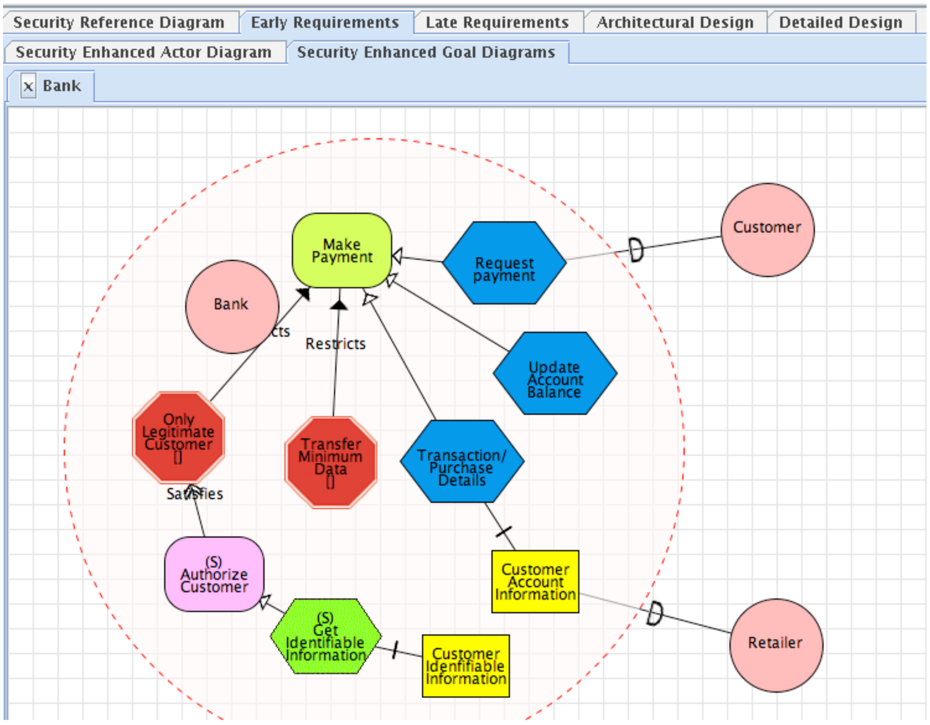


Fig. 9. Goal diagram

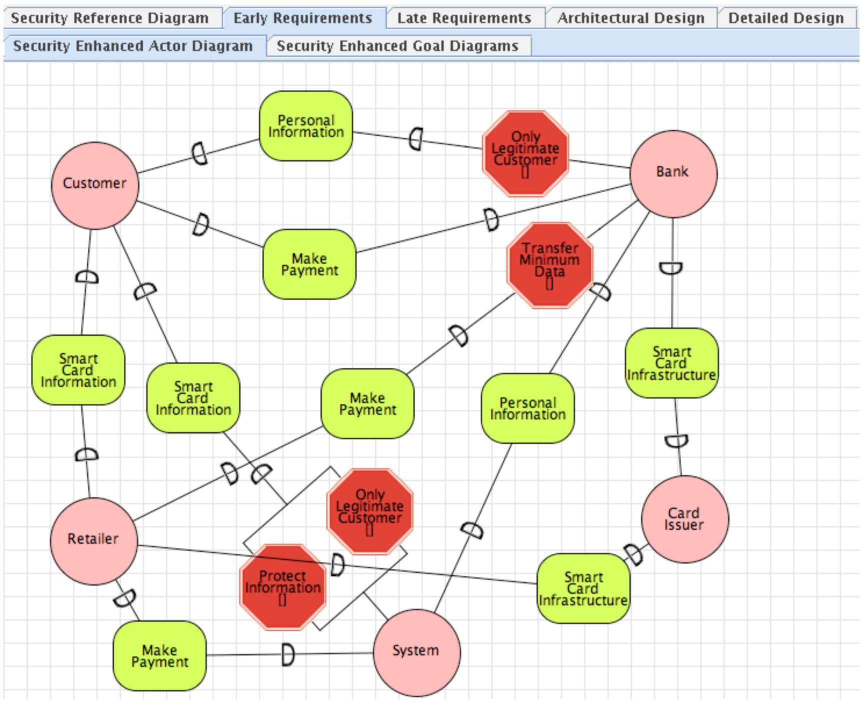


Fig. 10. Actor diagram with the system

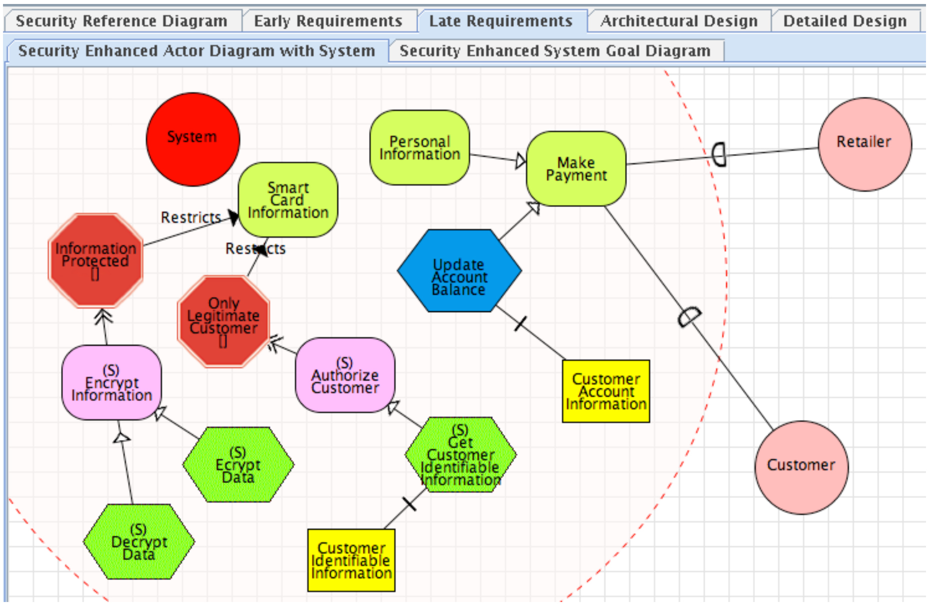


Fig. 11. Goal diagram of the System

During the architectural design the architecture of the system is defined. The tool enables the designer to construct an architecture style selection diagram where he can model different architecture styles and then model how much these architecture styles contribute to the security requirements that has identified in the previous stages.

Finally, in most cases, during the end of the architectural design the security attack testing takes places, where the design of the system is tested against the security requirements [11]. The tool automatically generates for the designer the security attack scenario template and the security test case template where the designer can model the attack and the countermeasures (Fig. 12). In the context of Secure Tropos, Security Attack Scenario is defined *as an attack situation describing the actors of a software system and their secure capabilities as well as possible attackers and their goals, and it identifies how the secure capabilities of the system's actors prevent (if they prevent) the satisfaction of the attackers' goals.*

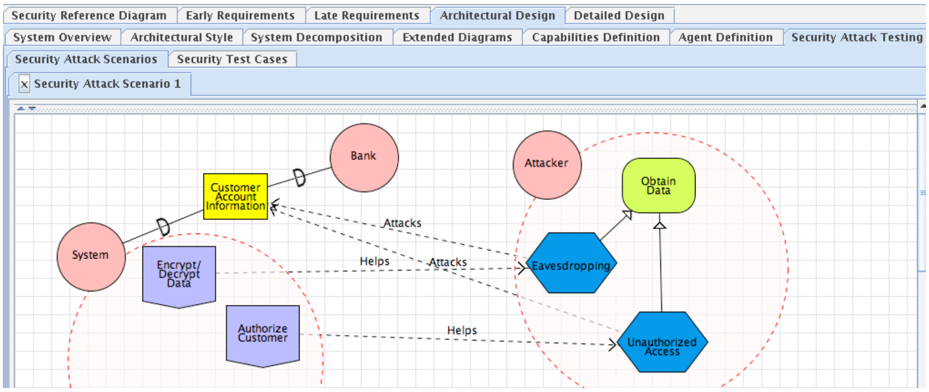


Fig. 12. Security attack scenario template

3.4 SecTro Additional Functionalities

Apart from the above-mentioned functionalities the tool provides additional functionalities such as checking the correctness of a model. The designer has the ability to draw models, but this includes the danger that designers might draw models that are not syntactically correct. Thus, the tool prevents a drawing action by the designer that is not complying with the syntax of Secure Tropos based on the meta-model and shows a respective warning notification. Therefore, a specific concept within a model cannot link with another concept unless rules are defined within the meta-model. This enables the tool to check the correctness of the individual model. For example if the developer tries to connect a security constraint with a goal with a "contributes" link, the tool will not allow such action because syntactically a security constraint connects with a "restricts" link with the goal. Furthermore, the tool checks the consistency between models. When changes are made in one model they are automatically reflected in the other models. For example, when a name of an actor is changed in the early requirements diagrams it is automatically changed in the diagrams of the late requirements.

Complementary to the above-mentioned functionalities is the functionality of the SecTro Assistant. It is located at the bottom of the workspace and it has been developed to assist designer that are not familiar with Secure Tropos methodology. SecTro Assistant provides feedback on designers' actions, so for example, when a specific toolbox button or tab is clicked, it is showing detailed information about the concept of the button or about the modelling activity of the stage tab. In addition, when the tool has prevented an action by enforcing rule checking on the models, SecTro Assistant will show more information about the reason behind the prevention of the action. Also, by clicking a button in the SecTro Assistant the designer can see parts of the Secure Tropos meta-model that are related to his action.

Furthermore, to increase the interoperability of the tool, the tool includes a functionality that enables the designer to export the developed models in XML format. The output is a .XML file that contains the model in XML format and can be used as an input to another tool.

Last but not least functionalities that SecTro supports are the designer's options to export a model as an image, to print a model, and to customize the workspace. In case of exporting a model as an image the designer has the option to choose the extension of the output image file according to his preference. While in the case of workspace customization, the designer has the option to zoom in and out of the model, to turn the grid on or off, to change the background colour of his model, and to hide the panel around the drawing canvas in order to create more space for his model development.

Therefore, we believe the tool is capable to produce models that are required by the designer during the secure software development process. It provides a usable interface with all the concepts that the designer will need when using the Secure Tropos methodology.

4 Related Work

Although Secure Tropos is still in research and it is difficult to develop a CASE tool for a methodology that is still in research, the i* modelling framework has been out for some years and a number of related CASE tools were developed to support it. OME [12] is a general, goal or agent oriented modelling and analysis tool and its improved version, OpenOME [13], additionally supports aspect oriented engineering while it is integrated in Eclipse Integrated Development Environment (IDE), in Protégé conceptual modelling environment and other graph editing environments such as Microsoft Visio. REDEPEND [14] is a graphical modelling tool that allows the developer to build and analyze Strategic Dependency (SD) models and Strategic Rationale (SR) models and is a plug-in for Microsoft Visio. The REDEPEND-REACT [14] tool is an extension of REDEPEND and involves the ability to define properties, such as security and efficiency. Also, following some heuristics rules, it guides the process of generating an architecture by recommending specific ones. Then the candidate architectures can be evaluated against a variety of properties. TAOM4e

[15] is also a plug-in for the Eclipse IDE and supports the modelling in tropos methodology during all its phases and also generates automatically the code from the tropos specification to JADE or jadex implementations. This can be achieved by mapping the tropos meta-model concepts to the target implementation language constructs. The GR-Tool [16] is a graphical tool, where the developer can construct goal models and run the algorithms that are embedded in the tool for forward and backward reasoning. The T-Tool [17] performs model checking of Tropos specifications while the ST-Tool [17] is a tool to support an extension of tropos that provides trust management process and performs formal analysis of its specifications. J-PRiM [18] tool is a plug-in for Eclipse IDE that uses the i* modeling and supports the J-PRiM methodology, which is a re-engineering methodology, where the specification of the new system starts with the observation of the current system and ends with the specification of the system to be. SNet Tool [19] provides an automatic transformation of graphical network representations based on the extended i* into executable programs. As a result, network scenarios can be simulated and provide valuable feedback. HeRA is a tool that incorporates the concept of heuristic requirements elicitation and uses for elicitation and analyse of security requirements [4]. HeRA uses heuristic rules to represent security related experience and extend further to capture the organisational knowledge for identifying security requirements [5].

The aforementioned tools, although they were developed for different ultimate purposes, they provide support for the i* modelling framework, which is the modelling framework that was adopted by Secure Tropos as well. But, Secure Tropos introduces new concepts that none of the previous tools enables their graphical representation, i.e. security constraint, secure goal, secure plan, secure resource, and secure capability. Also, the previous tools don't provide support for the modelling activities that Secure Tropos introduces, i.e. security constraint modelling, secure entities modelling, and secure capability modelling. So, despite the fact that experienced users with Secure Tropos can make conventions and use the previous tools to construct single diagrams; these tools are not adequate to support the Secure Tropos methodology.

5 Conclusions and Future Work

This paper presented our effort to develop a CASE tool for the Secure Tropos methodology. The tool supports the designers for the security modelling activities in particular for elicitation and analysis of early and late requirements. SecTro aims to provide an editor to develop the security models based on Secure Tropos. It has a user-friendly interface, which makes it easy to use and assists analysts who are not familiar with the methodology, by providing them with information about the methodology concepts, stages, and metamodels. Also, it enforces rules and constraints and provides valuable feedback on various actions of the designers in an interactive way. The tool has already been used by the students of university of East London to model and analyse security issues of a real industry case study. However, the tool does not support the modelling activities of the detailed design stage and we consider this as future work. In addition, future work includes the extension of the XML Schema in order to validate more models of the methodology.

References

1. Islam, S., Mouratidis, H., Jürjens, J.: A Framework to Support Alignment of Secure Software Engineering with Legal Regulations. Journal of Software and Systems Modeling (SoSyM), Theme Section on Non-Functional System Properties in Domain-Specific Modeling Languages (NFPinDSML) 10(3), 369–394 (2011)
2. Islam, S., Mouratidis, H., Wagner, S.: Towards a Framework to Elicit and Manage Security and Privacy Requirements from Laws and Regulations. In: Wieringa, R., Persson, A. (eds.) REFSQ 2010. LNCS, vol. 6182, pp. 255–261. Springer, Heidelberg (2010)
3. Mouratidis, H., Giorgini, P.: Integrating Security and Software Engineering: Future Vision and Challenges. In: Mouratidis, H., Giorgini, P. (eds.) Integrating Security and Software Engineering: Advances and Future Visions. Idea Group Publishing, London (2007)
4. Houmb, S.H., Islam, S., Knauss, E., Jürjens, J., Schneider, K.: Eliciting Security Requirements and Tracing them to Design: An Integration of Common Criteria, Heuristics, and UMLsec. Requirements Engineering Journal 15(1), 63–93 (2010)
5. Schneider, K., Knauss, E., Houmb, S.H., Islam, S., Jürjens, J.: Enhancing Security Requirements Engineering by Organisational Learning. Requirements Engineering Journal (REJ), Special Issue on REFSQ (2011)
6. Mouratidis, H., Giorgini, P.: Secure Tropos: A Security-Oriented Extension of the Tropos Methodology. International Journal of Software Engineering and Knowledge Engineering 17(2), 285–309 (2007)
7. Giorgini, P., Mouratidis, H., Zannone, N.: Modelling Security and Trust with Secure Tropos. In: Mouratidis, H., Giorgini, P. (eds.) Integrating Security and Software Engineering: Advances and Future Visions. Idea Group Publishing, London (2007)
8. Matulevicious, R.: Summary of Secure Tropos Metamodel. Internal Report, University of Namur (2008)
9. Pavlidis, M., Islam, S.: SecTro: A CASE Tool for Modelling Security in Requirements Engineering using Secure Tropos. In: Nurcan, S. (ed.) Proceedings of the Conference on Advanced Information Systems Engineering (CAiSE) Forum, London, pp. 89–96 (2011)
10. SecTro | Homepage, `http://sectro.securetropos.org/`
11. Mouratidis, H., Giorgini, P.: Security Attack Testing (SAT) – Testing the Security of Information Systems at Design Time. Journal of Information Systems 32, 1166–1183 (2007)
12. OME3, `http://www.cs.toronto.edu/km/ome/`
13. OpenOME, `https://se.cs.toronto.edu/trac/ome/`
14. Grau, G., Franch, X., Maiden, N.: REDEPEND-REACT: An Architecture Analysis Tool. In: 13th IEEE International Conference on Requirements Engineering, Paris, pp. 455–456 (2005)
15. Morandini, M., Nguyen, D.C., Perini, A., Siena, A., Susi, A.: Tool-Supported Development with Tropos: The Conference Management System Case Study. In: Luck, M., Padgham, L. (eds.) AOSE 2007. LNCS, vol. 4951, pp. 182–196. Springer, Heidelberg (2008)
16. Giorgini, P., Mylopoulos, J., Sebastiani, R.: Goal-Oriented Requirements Analysis and Reasoning in Tropos Methodology. Journal of Engineering Applications of Artificial Intelligence 18(2), 159–171 (2005)

17. Giorgini, P., Massacci, F., Mylopoulos, J., Zannone, N.: ST-Tool: A CASE Tool for Security Requirements Engineering. In: 13th IEEE International Conference on Requirements Engineering, Paris, pp. 451–452 (2005)
18. Grau, G., Franch, X., Avila, S.: J-PRiM: A Java Tool for a Process Reengineering i* Methodology. In: 14th IEEE International Conference on Requirements Engineering, Minneapolis, pp. 359–360 (2006)
19. Gans, G., Lakemeyer, G., Jarke, M., Vits, T.: SNet: A Modeling and Simulation Environment for Agent Networks Based on i* and ConGolog. In: Pidduck, A.B., Mylopoulos, J., Woo, C.C., Ozsu, M.T. (eds.) CAiSE 2002. LNCS, vol. 2348, pp. 328–343. Springer, Heidelberg (2002)

A Tool for Managing Evolving Security Requirements*

Gábor Bergmann[1], Fabio Massacci[2], Federica Paci[2], Thein Than Tun[3], Dániel Varró[1], and Yijun Yu[3]

[1] DMIS - Budapest University of Technology and Economics
`{bergmann,varro}@mit.bme.hu`
[2] DISI - University of Trento
`{fabio.massacci,federica.paci}@unitn.it`
[3] Department of Computing - The Open University
`{t.t.tun,y.yu}@open.ac.uk`

Abstract. Management of requirements evolution is a challenging process. Requirements change continuously making the traceability of requirements difficult and the monitoring of requirements unreliable. Moreover, changing requirements might have an impact on the security properties a system design should satisfy: certain security properties that are satisfied before evolution might no longer be valid or new security properties need to be satisfied after changes have been introduced. This paper presents SeCMER, a tool for requirements evolution management developed in the context of the SecureChange project. The tool supports automatic detection of requirement changes and violation of security properties using change-driven transformations. The tool also supports argumentation analysis to check security properties are preserved by evolution and to identify new security properties that should be taken into account.

Keywords: security requirements engineering, secure i*, security argumentation, change impact analysis, security patterns.

1 Introduction

Modern software systems are increasingly complex and the environments where they operate are increasingly dynamic. The number and needs of stakeholders are also changing constantly as they adjust to changing environments. A consequence of this trend is that the requirements for a software system are many and they change continuously. To deal with evolution, we need analysis techniques that assess the impact of system evolution on the satisfaction of requirements. Requirements for system security, in particular, are very sensitive to evolution: security properties satisfied before the evolution might no longer hold or new security properties need to be satisfied as result of the evolution.

* Work partly supported by the project EU-FP7-ICT-FET-IP-SecureChange.

Another important aspect is that change management process is a complex process that would benefit from tool support. However, changes make the traceability of requirements difficult and the monitoring of requirements unreliable: requirements management is time-consuming and error-prone when done manually. Thus, a semi-automated requirements evolution management environment, supported by a tool, will improve requirement management with respect to keeping requirements traceability consistent, realizing reliable requirements monitoring, improving the quality of the documentation, and reducing the manual effort.

In this paper we present SeCMER[1], a tool developed in the context of the SecureChange European project[2]. The tool supports the different steps of SeCMER methodology for evolutionary requirements [10]. The methodology supports the automatic detection of requirement changes and violation of security properties, and argumentation analysis [16] to check security properties are preserved by evolution and to identify new security properties that should be taken into account.

In the next section we give an overview of the SeCMER methodology; then in Sec. 3 we describe the tool architecture. In Sec. 4 we illustrate the tool features based on an industrial example of evolution taken from the air traffic management domain. After presenting related works in Sec. 5, the results of tool evaluation are discussed in Sec. 6. Finally Sec. 7 concludes the paper.

2 SeCMER Methodology

The SecureChange Methodology for Evolutionary Requirements (SeCMER) [10] supports:

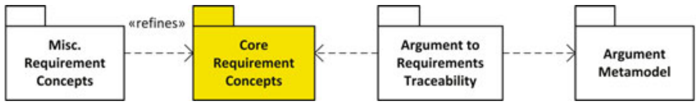
– a conceptual model of security requirements and a process for the elicitation of security goals
– a light-weighted approach to formalizing and reasoning about changing security goals, and
– an approach based on argumentation and model transformation to reason about the impact of change.

The main output from the methodology is either an assurance that the changes did not make the system violate the existing security properties, or a formulation of new properties to be satisfied by the new design. In the next subsections we will illustrate the main steps of the SeCMER methodology.

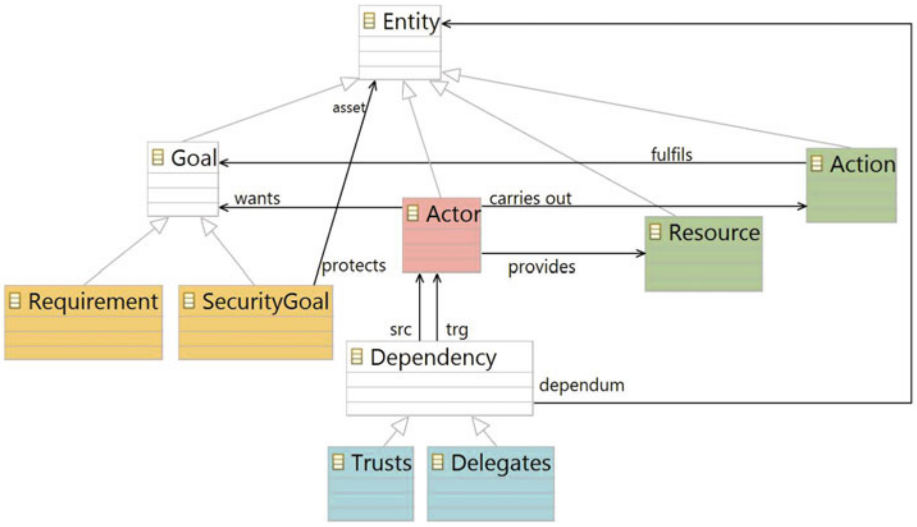
2.1 Modeling of Evolving Requirements

The Modeling of Evolving Requirements step produces two requirements model the *before model* and the *after model* which are an instance of the SeCMER conceptual model [13]. The conceptual model identifies a set of core concepts

[1] A detailed description of the tool implementation is reported in [11].
[2] www.securechange.eu


Misc. Requirement Concepts
«refines»
Core Requirement Concepts
Argument to Requirements Traceability
Argument Metamodel

(a) Package-level overview (extract)



(b) Package of core SeCMER concepts (extract)

Fig. 1. SeCMER conceptual model

that link the empirical security knowledge such as information about assets, security goals and threats to the stakeholders' security goals. To create this link, the conceptual model amalgamates concepts from Problem Frames (PF) [12] and SI* requirements engineering methodology [14] with traditional security concepts such as security goal and asset.

SI* [14] extends the i* framework which allows to model the stakeholders for a given project, their goals and their social inter-dependencies. In SI*, actors have goals, and own resources and tasks. The Problem Frames [12] approach (PF) instead explores the relationship between the machines, the physical domains in the problem world context and the requirements. Concepts from Problem Frames diagrams are similar to SI* concepts. For instance, the notion of biddable domain is similar to the notion of actor. Other types of domains such as lexical domains and causal domains are similar to resource and asset. The notion of phenomena in Problem Frames is generic enough to cover action, event and state.

The combination of the two security goals engineering approaches has several advantages: with SI* analysis, malicious intentions of attackers can be identified through explicit characterization of social dependencies among actors; with PF security goals analysis, valuable assets that lie within the system boundary can be identified through explicit traceability of shared phenomena among physical domains and the machine itself.

The SecMER conceptual model is illustrated in Figure 1. Figure 1 b) shows the core requirement concepts that are relevant for change detection and security analysis based on argumentation:

– An *actor* is an entity that can act and intend to want or desire something.
– An *action* is a means to achieve a goal.
– A *resource* is an entity without intention or behavior, and can be provided by actors.
– An *asset* is any entity of value for which protection is required.
– A *goal* is a proposition an actor wants to make true.
– A *requirement* is a goal that could be satisfied by a software system.
– A *security goal* is a goal to prevent harm to an asset through the violation of security properties.

The conceptual model also includes a set of relationships between concepts which include do-dependency, can-dependency and trust-dependency adopted from SI*. The *protects* relationship is a relationship between a security goal and a resource, action or goal – that denotes they are assets. For a complete list of all the possible relationships supported in SeCMER conceptual model the reader is referred to [13].

2.2 Change Detection Based on Security Patterns

The SeCMER methodology includes a lightweight automated analysis step that evaluates requirements-level compliance with security principles. These security principles are declaratively specified by an extensible set of *security patterns*.

A security pattern expresses a situation (a graph-like configuration of model elements) that leads to the violation of a security property. Whenever a new match of the security pattern (i.e. a new violation of the security property) emerges in the model, it can be automatically detected and reported. The specification of security patterns may also be augmented by automatic remedies (i.e. templates of corrective actions) that can be applied in case of a violation to fix the model and satisfy the security property once again.

SeCMER includes extension facilities that allow plug-ins to contribute the declarative definition of *security pattern*s in a high-level change-driven language [9] based on the notion of graph patterns. Automated solution templates (defined programmatically) can also be contributed. The tool then detects violations of these security patterns, which will appear as problem markers (warnings). The suggested solutions appear as Quick Fix rules offered for the problem marker.

Although the set of security patterns is extensible, the main focus points of security patterns are the following: *trust* (which can be explicitly modeled, and interpreted transitively), *access* (which can also be granted / delegated transitively), and *need* (expressed by carrying out an action that consumes a resource). The patterns are further characterised by the following:

- The security patterns only consider *assets* that are protected by security goals.
- If a trusted actor performs an action that is known to fulfill the security goal, then no further investigation is required.
- If there is access to an asset without trust (regardless of need), then it is considered a violation of the *trusted path* property.
- If there is access to an asset without the need thereof (regardless of trust), then it is considered a violation of the *least privilege* property.
- If there is need for an asset but no actual access, then the model is reported as inconsistent / incomplete.
- Security violation reports can be suppressed by manual arguments supporting the satisfaction of the security goal.

Example: The *trusted path* security pattern finds security violations where an asset is communicated via an untrusted path. The pattern has the following structure: if a concerned actor wants a security goal that expresses that a resource must be protected, then each actor that the resource is delegated to must be trusted (possibly transitively) by the concerned actor. An exception is made if a trusted actor performs an action to explicitly fulfill the security goal, e.g. digital signature makes the trusted path unnecessary in case of an integrity goal.

See Lst. 1 for the simplified definition of the pattern using the declarative model query language of EMF-IncQuery [8]. According to the pattern definition, a violation of the trusted path property is characterized by a quadruplet of model elements *ConcernedActor*, *SecGoal*, *Asset*, *UntrustedActor*, provided that they satisfy a list of criteria (graph constraints listed between the pair of braces). Regarding the type and configuration of these elements, as enforced by the two edge and one node constraint in lines 2-4, *SecGoal* is an instance of type Security Goal that is wanted by an actor *ConcernedActor* and expresses the protection of the element *Asset*. Lines 5-6 state that *Asset* is provided by some actor *ProviderActor* (not exposed as a parameter of the pattern), and - through a chain of transitive delegation - is eventually possessed by the *UntrustedActor*; thanks to the pattern composition language feature, the latter is expressed by a helper pattern *transitiveDelegation* (defined elsewhere). A negated condition on line 7 ensures that *UntrustedActor* is not trusted (transitively) by the *ConcernedActor*. A second negative constraint (line 8) expresses that the *SecGoal* is not fulfilled explicitly by any action that is trusted by the *ConcernedActor*.

The security pattern in Lst. 1 can be applied to enforce a security property such as integrity. Requirements engineers are further assisted by a set of suggested fixes that can be applied on violations of the security property. In fact, each of these suggestions can be implemented as automated corrective actions to be applied to the model in order to re-establish the security property. The requirements engineers can then choose one of the suggestions, or come up with their own solution. Possible examples of corrective actions include:

Listing 1. Pattern to capture violations of the trusted path property

```
shareable pattern untrustedPath(ConcernedActor,SecGoal,Asset,UntrustedActor)={
    Actor.wants(ConcernedActor,SecGoal);
    SecurityGoal(SecGoal);
    SecurityGoal.protects(SecGoal, Asset);
    Actor.provides(ProviderActor,Asset);
    find transitiveDelegation(ProviderActor,UntrustedActor,Asset);
    neg Actor.trust*(ConcernedActor,UntrustedActor);
    neg find trustedFulfillment(ConcernedActor,AnyActor,AnyTask,SecGoal);
}
```

– Add a trust relationship from *ConcernedActor* to *UntrustedActor* to reflect that the security decision was that there must be trust between these actors (e.g. by establishing a liability contract between them).
– Alternatively, an action can be created that explicitly fulfills *SecGoal*, such as introducing a policy or technological process that makes it impossible for *UntrustedActor* to abuse the situation (e.g. digital signature to ensure the security goal of data integrity).

These solution templates can be attached to the security pattern so that they are offered whenever a violation of the corresponding security property is detected. The solutions can be implemented by arbitrary program code, typically short snippets that manipulate the model according to the description of the solution.

2.3 Argumentation-Based Security Analysis

In this step of the SeCMER methodology, the developers check whether there are new security properties to be added or to be removed (Δ Security Properties) as a result of changes in the requirement model. This phase is supported by argumentation analysis.

As shown in the meta-model of the SeCMER arguments in Figure 2, an argument diagram may have several arguments linked to each other. An *argument* contains one and only one claim. It also contains facts and warrants. A *claim* is a predicate whose truth-value will be established by an argument. A *fact* is a true proposition (an argument with a claim only). A *warrant* links facts in an argument to the claim. Since facts and warrants can themselves be arguments, arguments can be nested. Every argument has an optional timestamp, which indicates the time (or the round) during the argumentation process at which the argument is introduced.

As well as nesting of arguments, arguments may be related to each other through rebuttal and mitigation/restore relationships. A rebuttal argument is a kind of argument whose purposes are to establish the falsity of their associated argument or make them inconsistent. Similarly, mitigations are another special kind of arguments following the iteration of rebuttals in order to reestablish the truth-value of the associated original claims. Mitigations may or may not negate the claims of the rebuttals: sometimes they add further facts overlooked by the rebuttals.

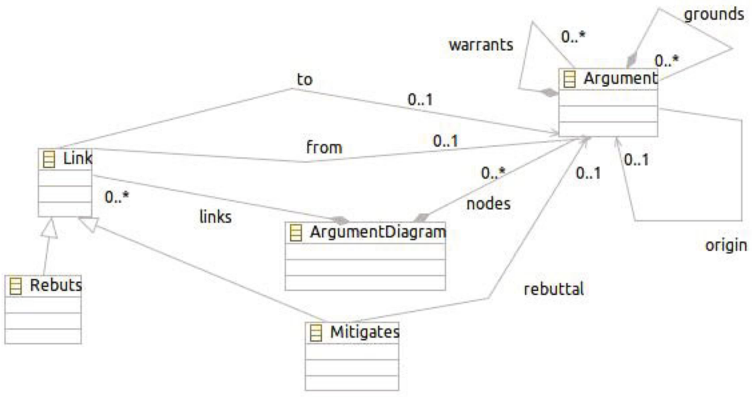


Fig. 2. Meta-Model of SeCMER Arguments

Figure 3 illustrates the visual syntax of SeCMER argument diagrams. Graphically, an argument is represented by a box with three compartments: the claim is written in the top compartment, the fact(s) in the middle compartment and the warrant(s) in the bottom compartment. Rebuttal and mitigation links are represented by the red and green arrows respectively.

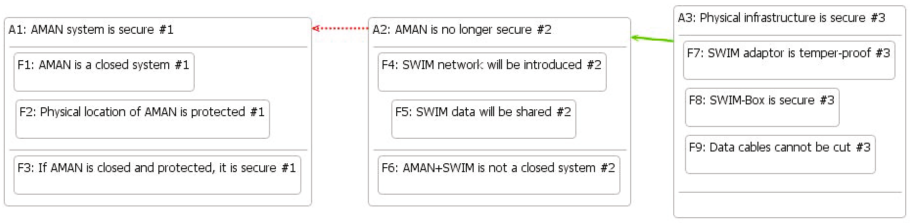

A1: AMAN system is secure #1
F1: AMAN is a closed system #1
F2: Physical location of AMAN is protected #1
F3: If AMAN is closed and protected, it is secure #1
A2: AMAN is no longer secure #2
F4: SWIM network will be introduced #2
F5: SWIM data will be shared #2
F6: AMAN+SWIM is not a closed system #2
A3: Physical infrastructure is secure #3
F7: SWIM adaptor is temper-proof #3
F8: SWIM-Box is secure #3
F9: Data cables cannot be cut #3

Fig. 3. Visual Syntax of SeCMER Arguments

Since argumentation is a costly manual process, it is preferable to avoid its re-execution after each small change of the requirement model. However, some arguments may be invalidated by evolution and require attention from security experts. Therefore, if a change affects one of the elements that was recorded as an evidence for an argument, then the argument is marked for re-examination. This relies on traceability links that can be established between the argument and requirement models.

3 SeCMER Tool Architecture and Implementation

SeCMER is an Eclipse-based heterogeneous modeling environment for managing evolving requirements models. It has the following features:

– **Modeling of Evolving Requirements**. Requirement models can be drawn in SI*, Problem Frames or SeCMER. Traceability and bidirectional synchronization is supported between SeCMER and SI* requirements models.
– **Change detection based on security patterns**. Violations of formally defined static security properties expressed as security patterns can be automatically identified. Detection of formal or informal arguments that has been invalidated by changes affecting model elements that contributed to the argument as evidence is also supported.
– **Argumentation-based security analysis**. Reasoning about security properties satisfaction and identification of new security properties is supported.

These capabilities of the tool are provided by means of the integration of a set of EMF-based [15] Eclipse plug-ins written in Java, relying on standard EMF technologies such as GMF, Xtext and EMF Transaction. The components of the tool are:

– Eclipse plug-ins of OpenPF including (a) the implementation of the SecMER conceptual model, (b) the argumentation model and tools, as well as (c) external modeling tools for Problem Frames [17],
– SI* (requirements engineering tool [14]),
– traceability models to represent the relationship between corresponding model elements in different languages, e.g. the SecMER conceptual model and SI*,
– run-time platform components of EMF-IncQuery (incremental EMF model query engine) for change-driven transformations,
– model query plug-ins automatically generated from (a) security patterns or (b) transformation specification by the development-time tools of EMF-IncQuery,
– integration code developed solely for this tool, including User Interface commands and the Java definition of the action parts of (a) quick fixes and (b) model synchronization.

The relationship of the most important model management components are depicted on Fig. 4, focusing on the SI* and SeCMER models in particular, as well as the traceability model established between them. User Interface components are omitted from this diagram.

All the involved EMF models are accessed through a common EMF `ResourceSet` and edited solely through the corresponding `TransactionalEditingDomain` (from the EMF Transaction API). Consequently, all modifications are wrapped into EMF Transactions, including those carried out by manual editing through the User Interface (e.g. the SI* diagram editor) as well as changes performed by automated mechanisms such as model transformation. As one of the benefits, concurrent modifications are serialized and therefore conflict-free. Furthermore, the commit process of the transactions provides an opportunity for triggering change-driven actions.

The incremental query mechanism provided by EMF-IncQuery plays a key role in the functionality of the tool. Incremental query evaluation code is generated automatically at development time by EMF-IncQuery, from a graph

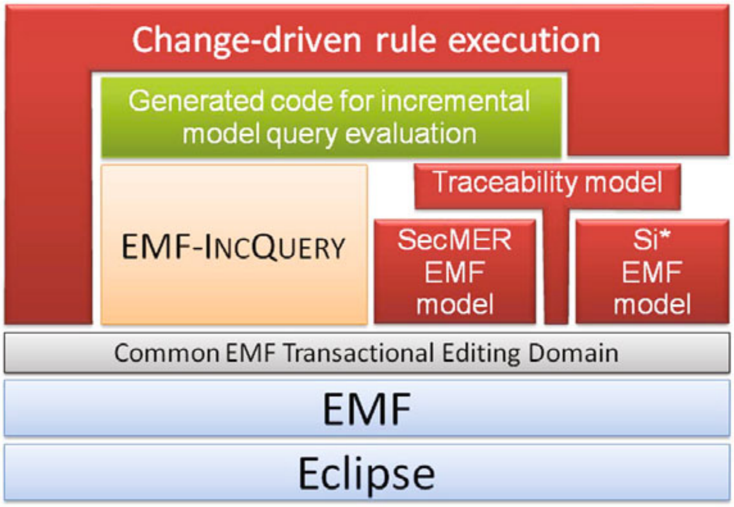

Change-driven rule execution
Generated code for incremental model query evaluation
EMF-IncQuery
Traceability model
SecMER EMF model
Si* EMF model
Common EMF Transactional Editing Domain
EMF
Eclipse

Fig. 4. Architectural overview of model management components

pattern-based declarative description of EMF model queries. Through this incremental evaluation functionality, change-driven rules can be efficiently triggered by changes captured as graph patterns. The implementation currently supports detecting the presence, appearance and disappearance of graph patterns. A more advanced formalism for capturing changes is already defined [9], but support is not implemented yet.

The core trigger engine plug-in offers an Eclipse extension point for defining change-driven rules. Multiple constituent plug-ins contribute extensions to register their respective set of rules. The graph pattern-based declarative guard of the rules is evaluated efficiently (see measurements in [8]) by the incremental graph pattern matcher plug-ins automatically generated from the declarative description by EMF-IncQuery. At the commit phase of each EMF transaction, the rules that are found to be triggered will be executed to provide their reactions to the preceding changes. These reactions are implemented by arbitrary Java code, and they are allowed to modify the model as well (wrapped in nested transactions) and could therefore be reacted upon.

So far, there are three groups of change-driven rules as extension points:

– transformation rules that realize the on-the-fly synchronization between multiple modeling formalisms,
– security-specific rules that detect the appearance of undesired security patterns, raise alerts and optionally offer candidate solutions.
– rules for invalidating arguments when their ground facts change.

Another key feature is a bi-directional synchronizing transformation between SI* and the SeCMER model with changes propagated on the fly, interactively. Since the languages have different expressive power, the following challenges arise:

1. some concepts are not mapped from one formalism to the other or vice versa,
2. some model elements may be mapped into multiple (even an unbounded amount of) corresponding model elements in the other formalism, and finally
3. it is possible that a single model element has multiple possible translations (due to the source formalism being more abstract); one of them is created as a default choice, but it can later be changed to the other options, which are also tolerated by the transformation system.

4 Illustrative Example

We now illustrate the features supported by our tool using the ongoing evolution of ATM (Air Traffic Management) systems as planned by the ATM 2000+ Strategic Agenda [7] and the SESAR Initiative.

Part of ATM system's evolution process is the introduction of the Arrival Manager (AMAN), which is an aircraft arrival sequencing tool to help manage and better organize the air traffic flow in the approach phase. The introduction of the AMAN requires new operational procedures and functions that are supported by a new information management system for the whole ATM, an IP based data transport network called System Wide Information Management (SWIM) that will replace the current point to point communication systems with a ground/ground data sharing network which connects all the principal actors involved in the Airports Management and the Area Control Centers.

We have chosen to illustrate the following steps of the SeCMER methodology based on the above evolutionary scenario.

1. **Requirements evolution**. We show how SeCMER supports the representation of the evolution of the requirement model as effect of the introduction of the SWIM.
2. **Change detection based on security patterns.**
   a *Detection of a security property violation based on security patterns.* We show how the tool detects that the integrity security property of the resource MD "Meteo Data" is violated due to the lack of a trusted path.
   b *Automatically providing corrective actions based on security patterns.* We show how violations of the integrity security property, as detected by a security pattern, may have corrective actions associated with them.
3. **Argumentation-based security analysis**. We show how argumentation analysis [16] can be carried to provide evidence that the information access property applied to the meteo data is satisfied after evolution.

The entities involved in the simple scenario are the AMAN, the Meteo Data Center (MDC), the SWIM-Box and the SWIM-Network. The SWIM-Box is the core of the SWIM information management system which provides access via defined services to data that belong to different domain such as flight, surveillance, meteo, etc. The introduction of the SWIM requires suitable security properties to be satisfied: we will show how to protect information access on meteo data and how to ensure integrity of meteo data.

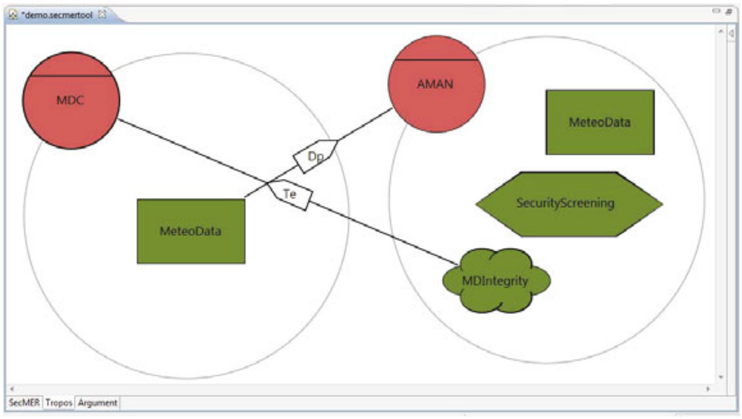


Fig. 5. Requirement Model before evolution (Si* syntax)

Requirements evolution. Figure 5 shows the before requirement model which consists of two actors the *AMAN* and *MDC*: MDC provides the asset Meteo Data (*MD*) to the AMAN. The AMAN has an integrity security goal *MDIntegrity* for MD, and MDC is entrusted with this goal. AMAN also performs an Action, *SecurityScreening*, to regularly conduct a background check on its employees to ensure that they do not expose to risk the information generated by the AMAN.

As the communication between the AMAN and MDC is mediated by the SWIM, the before model evolves as follows (see Figure 6):

– The Actors *SWIM*, *SWIMBox_ MDC* and *SWIMBox_ AMAN* are introduced in the SI* model
– As the meteo data is no longer directly provided by MDC to AMAN, the delegation relation between the two is removed.
– Delegation relationships are established between the Actors MDC, SWIMBox_MDC, SWIM, SWIMBox_AMAN, AMAN.
– As the SWIM network can be accessed by multiple parties, the AMAN has a new security goal *MDAccessControl* protecting MD resource.

Detecting violations of security properties based on security patterns. According to the pattern of Lst. 1, the integrity property for MD is violated because AMAN entrusts MDC with the integrity security goal, but not the communitation intermediary actors SWIMBox_MDC, SWIM and SWIMBox_AMAN. The violation (i.e. a match of the pattern) is detected and reported by the tool, as shown on Fig. 7.

Automatic corrective actions based on security patterns. The following quick fix suggestions are associated with the security pattern:

– Add a trust relationship between MDC and SWIM Network having the integrity security goal as dependum.

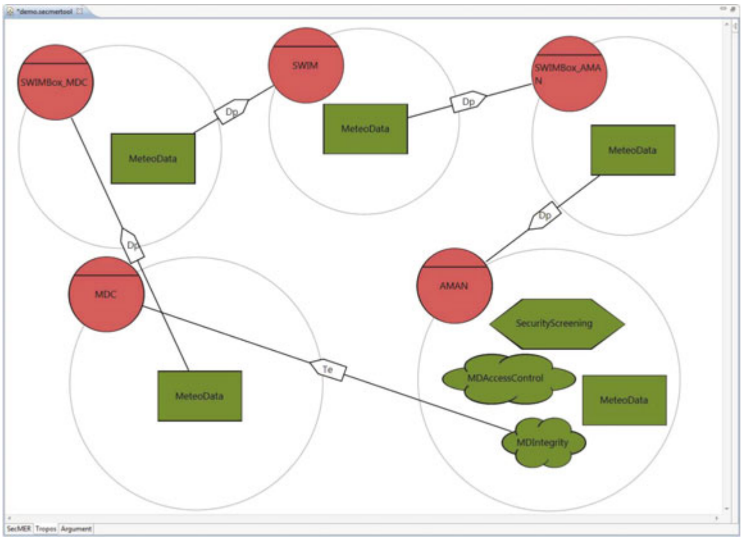
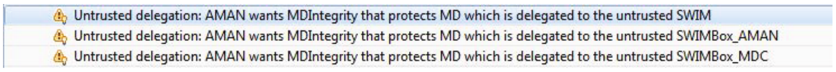
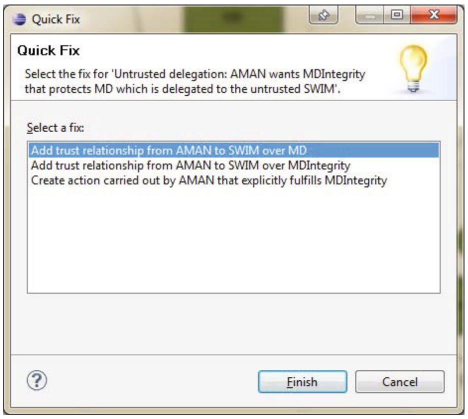


Fig. 6. Requirement after evolution (Si* syntax)


Untrusted delegation: AMAN wants MDIntegrity that protects MD which is delegated to the untrusted SWIM
Untrusted delegation: AMAN wants MDIntegrity that protects MD which is delegated to the untrusted SWIMBox_AMAN
Untrusted delegation: AMAN wants MDIntegrity that protects MD which is delegated to the untrusted SWIMBox_MDC

(a) Detected Security Issues



Quick Fix

Select the fix for 'Untrusted delegation: AMAN wants MDIntegrity that protects MD which is delegated to the untrusted SWIM'.

Select a fix:

Add trust relationship from AMAN to SWIM over MD
Add trust relationship from AMAN to SWIM over MDIntegrity
Create action carried out by AMAN that explicitly fulfills MDIntegrity

(b) Possible Corrective Actions

Fig. 7. Detection of Security Issues

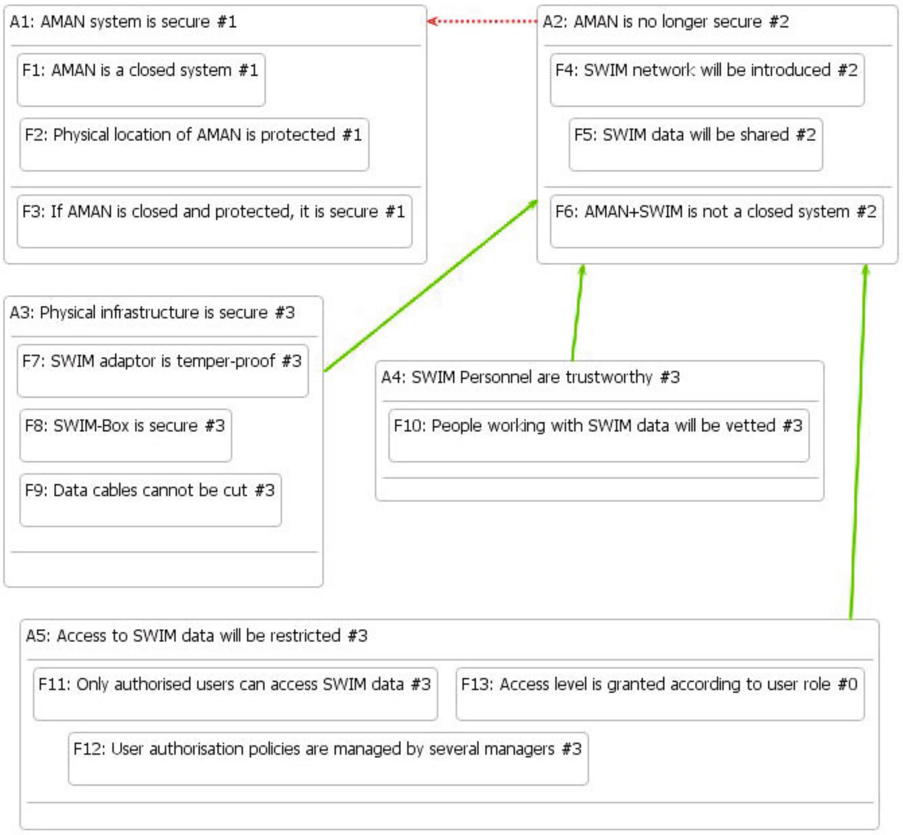

A1: AMAN system is secure #1
F1: AMAN is a closed system #1
F2: Physical location of AMAN is protected #1
F3: If AMAN is closed and protected, it is secure #1
A2: AMAN is no longer secure #2
F4: SWIM network will be introduced #2
F5: SWIM data will be shared #2
F6: AMAN+SWIM is not a closed system #2
A3: Physical infrastructure is secure #3
F7: SWIM adaptor is temper-proof #3
F8: SWIM-Box is secure #3
F9: Data cables cannot be cut #3
A4: SWIM Personnel are trustworthy #3
F10: People working with SWIM data will be vetted #3
A5: Access to SWIM data will be restricted #3
F11: Only authorised users can access SWIM data #3
F13: Access level is granted according to user role #0
F12: User authorisation policies are managed by several managers #3

Fig. 8. A fragment of the argumentation model

– Alternatively, an Action such as "MD is digitally signed" can be created to protect the integrity of MD even when handled by untrusted actors.

Argumentation for the information access property. Fig. 8 shows the different rounds of the argumentation analysis that is carried out for the information access security property applied to MD resource. The diagram says that the AMAN system is claimed to be secure before the change (Round #1), and the claim is warranted by the facts the system is known to be a close system (F1), and the physical location of the system is protected (F2). This argument is rebutted in Round #2, in which another argument claims that the system is no longer secure because SWIM will not keep AMAN closed. The rebuttal argument is mitigated in Round #3 by three arguments, which suggest that the AMAN may still be secure given that the physical infrastructure is secure, personnel are trustworthy and access to data is controlled.

5 Related Works

There are many requirement engineering tools available but only some of them support specific capabilities for requirement change management. CASE Spec [1] makes easy to generate traceability reports and perform impact analysis with built-in visual and tabular traceability tools. Dimensions RM [2] allows enterprises to effectively manage change in requirements during the project lifecycle. In particular, DimRM facilitates the understanding of the impact of requirement changes and the creation of reports on requirements definition, baselines, change impact, and traceability. IBM Rational DOORS [3] has powerful capabilities for capturing, linking, analyzing and managing changes to requirements and their traceability. IBM Rational RequisitePro [4] is a requirements management tool that incorporates a powerful database infrastructure to facilitate requirements organization, integration, traceability and analysis. Moreover, it provides detailed traceability views that display parent/child relationships and shows requirements that may be affected by upstream or downstream changes. MKS Integrity 2009 [5] provides reuse and requirements change management capabilities coupled with meaningful (and traceable) relationships to downstream code and testing assets, which ensure communication of change, conformance to requirements and compliance with applicable governance or regulations.

Reqtify [6] is an interactive requirement traceability and impact analysis tool which can trace requirement from system, program and project levels to the entire levels of software or hardware component development lifecycle.

Compared with the above tools, SecMER provides support to the requirement engineer for handling security related changes. The tool supports automatic detection of requirement changes that lead to violation of security properties using change-driven transformations and suggests possible corrective actions. The tool also supports argumentation analysis to check security properties are preserved by evolution and to identify new security properties that should be taken into account.

6 Tool Evaluation

The SecMER tool has been validated during a workshop with Air Traffic Management experts. We had a total of fifteen participants: four requirement analysts and eleven ATM experts who were air traffic controllers and the others were Deep Blue[3] consultants. The participants were divided in three groups. Each group has to first create the before and after requirement models for the illustrative scenario introduced in Sec. 4; then, check security violations and select a possible suggested quick fix; and build an argument model for the after requirement model. The domain experts were given wild cards to provide feedbacks related to the application of the methodology steps and on the usability and reliability of

[3] Deep Blue is a human factors, safety and validation consultancy providing solutions throughout industry and the public sector in the field of transportation (http://www.dblue.it/)

the tool. The validation session had a duration of one hour and thirty minutes. During the validation session each group was observed by a requirement analyst. At the end of the validation session the requirement analysts gave a questionnaire to be filled out by the participants. Useful feedbacks have been provided by the domain experts that have been used to improve the tool usability and reliability. Each group reported that was not clear how to create before and after models and how to maintain the history of changes. The experts suggested to have a guideline or a source of help that explains when to use the most critical concepts; and the possibility of saving in the same project the before and after models. These issues have been addressed since. Moreover, for the participants was confusing to have different views of the same model - SeCMER view and SI* view. Since the *protects* relationship is not part of the standard SI* conceptual model but only of the SeCMER conceptual model, the participants were required to switch from the SI* view to the SeCMER view and add the relationship to the model. In order to improve the usability of the tool, the *protects* information is now made part of SI*, and does not require manual effort from the final user; we have also named the SI* concepts in the palette of the SI* view as the mapped concepts in the SeCMER view to converge the terminologies of the two views. About the automatic detection of violation of security properties, the participants suggested that more guildelines should be given about the state of security modeling even when no violations are detected. The tool now guides the user in creating the first security goal, as well as in identifying the protected assets of security goals.

7 Conclusions

This paper has presented SeCMER, a tool for managing evolving requirements. As shown by the ATM-based illustrative scenario, the tool supports visual modeling of security requirements. Additionally, argument models can be constructed manually to investigate the satisfaction of security properties; the tool detects invalidated arguments if the requirements model evolves. Finally, the tool performs continuous and automatic pattern-based violation detection of security properties, with optional "quick fix" corrective actions.

We plan to extend the tool in order to support other sets of security patterns to automate the detection and handling of security violations in a wider range of application scenarios. We plan also to realize a tighter integration with additional modeling formalisms (Problem Frames) and industrial tools e.g DOORS-TREK.

References

1. CASE Spec, http://www.analysttool.com/
2. Dimensions RM, http://www.serena.com/products/rm/index.html
3. IBM Rational DOORS, http://www-01.ibm.com/software/awdtools/doors/
4. IBM Requisite Pro, http://www-01.ibm.com/software/awdtools/reqpro/
5. IMKS Integrity (2009), http://www.mks.com/

6. Reqtify, http://www.geensoft.com/en/article/reqtify
7. EUROCONTROL ATM Strategy for the Years 2000+ Executive Summary (2003)
8. Bergmann, G., Horváth, Á., Ráth, I., Varró, D., Balogh, A., Balogh, Z., Ökrös, A.: Incremental Evaluation of Model Queries over EMF Models. In: Petriu, D.C., Rouquette, N., Haugen, Ø. (eds.) MODELS 2010. LNCS, vol. 6394, pp. 76–90. Springer, Heidelberg (2010)
9. Bergmann, G., et al.: Change-Driven Model Transformations. Change (in) the Rule to Rule the Change. Software and System Modeling (2011) (to appear)
10. Bergmann, G., et al.: D3.2 Methodology for Evolutionary Requirements, http://www.securechange.eu/sites/default/files/deliverables/-%20Methodology%20for%20Evolutionary%20Requirements_v3.pdf
11. Bergmann, G., et al.: D3.4 Proof of Concept Case Tool, http://www.securechange.eu/sites/default/files/deliverables/D3.4tt%20Proof-of-Concept%20CASE%20Tool%20for%20early%20requirements.pdf
12. Jackson, M.: Problem Frames: Analyzing and structuring software development problems. ACM Press, Addison Wesley (2001)
13. Massacci, F., Mylopoulos, J., Paci, F., Tun, T.T., Yu, Y.: An Extended Ontology for Security Requirements. In: Salinesi, C., Pastor, O. (eds.) CAiSE Workshops 2011. LNBIP, vol. 83, pp. 622–636. Springer, Heidelberg (2011)
14. Massacci, F., Mylopoulos, J., Zannone, N.: Computer-aided support for secure tropos. Automated Software Engg. 14, 341–364 (2007)
15. The Eclipse Project: Eclipse Modeling Framework, http://www.eclipse.org/emf
16. Tun, T.T., et al.: Model-based argument analysis for evolving security requirements. In: Proceedings of the 2010 Fourth International Conference on Secure Software Integration and Reliability Improvement, SSIRI 2010, pp. 88–97. IEEE Computer Society, Washington, DC (2010)
17. Yu, Y., Tun, T.T.: OpenPF - The Open Requirements Engineering Lab, http://computing-research.open.ac.uk/trac/openre

Tool Support for Enforcing Security Policies on Databases

Jenny Abramov[1,2], Omer Anson[2], Arnon Sturm[1], and Peretz Shoval[1]

[1] Department of Information Systems Engineering
[2] Deutsche Telekom Laboratories (T-Labs),
Ben-Gurion University of the Negev,
Beer Sheva 84105, Israel
{jennyab,sturm,shoval}@bgu.ac.il,
oaanson@gmail.com

Abstract. Security in general and database protection from unauthorized access in particular, are crucial for organizations. It has long been accepted that security requirements should be considered from the early stages of the development process. However, such requirements tend to be neglected or dealt-with only at the end of the development process. The Security Modeling Tool presented in this paper aims at guiding and enforcing developers, in particular database designers, to deal with database authorization requirements from the early stages of the development process. In this paper we demonstrate how the Security Modeling Tool assists the various stakeholders in designing secure database code and describe the tool architecture.

Keywords: Secure software engineering, database design, authorization.

1 Introduction

Data is the most valuable asset for an organization as its survival depends on the correct management, security, and confidentiality of the data [1]. In order to protect the data, organizations must secure data processing, transmission and storage. Developers of data-oriented systems always face problems related to security. Yet, these types of problems are usually ignored in the early stages of the development process.

In the last decade various methods were suggested to incorporate security aspects within the development process. Several UML security-related extensions were proposed, such as UMLsec [19] and SecureUML [17, 15]. Additionally, a security-oriented extension to the Goal-Driven Requirements Engineering methodology Tropos was proposed - Secure Tropos [18]. Mouratidis and Jurjens combined Secure Tropos and UMLsec [24] to create a structured methodology for secure software development that supports all software development phases. Fernández-Medina and Piattini [23] also proposed a method to design secure databases. Another approach for security specification is security patterns, which is based on the classic idea of design patterns introduced by the Gang of Four [20]. Security patterns were proposed to assist developers to handle security concerns and provide guidelines to be used from the early stages of the development lifecycle [5]. However, to successfully utilize a security pattern, there must be systematic guidelines supporting its application throughout the entire software development lifecycle. Such a methodology to build secure systems using patterns was presented by Schumacher et al. [25] and Fernandez et al. [16]. This methodology integrates security patterns into each one of the software development stages, and each stage can be tested for compliance with the principles presented by the patterns. A catalog of security patterns can help to define the security mechanisms at each architectural level and at each development stage. Hafner and Breu [21] proposed a model driven security methodology for service-oriented architectures. Other methodologies present the use of aspect-oriented software design to model security as separate aspects which would later be weaved within the functional model. For example, in [22] the authors propose to deal with access control requirements while utilizing UML diagrams.

The above studies, and other related studies (which are not referenced here due to space limit), mainly provide guidelines regarding the way security should be handled within certain stages of the software development process, or address specific aspects of security. To the best of our knowledge, no existing method provides a complete framework that both guides and enforces organizational security policies on a system design, and then generates executable code from that design.

To overcome these deficiencies, we have developed a methodology that enables organizations to specify their security policies in the form of security patterns, which will guide developers in the incorporation of these particular organizational security policies, as well as verifies their correct application. In addition, the methodology enables the developer to transform the result into code, based on the organizational policies. In this paper, we explicitly refer to the application of access control in databases.

The methodology incorporates ideas from two areas of expertise: in the area of *system development methodologies*, we adopt the principle of integrating data and functional modeling at the early stages of the development, according to the Functional and Object-Oriented Methodology (FOOM) [6]; in the area of *domain engineering*, we adopt the principles suggested by the Application Based Domain Modeling (ADOM) approach [4]. ADOM supports building reusable assets on the one hand, and representing and managing knowledge in specific domains on the other hand. This knowledge guides the development of various applications in that domain and serves as a verification template for their correctness and completeness.

The developed methodology is supported by the Security Modeling Tool (SMT), which enables the modeling of security patterns and enforces their correct usage during application development. The knowledge captured in the security patterns is used to automatically verify that the application models are indeed secure with respect to the defined patterns. Having a verified model, a secure database code can be automatically generated.

SMT is an Eclipse plug-in and is based on existing frameworks such as the Eclipse Modeling Framework [2], which is used to interface with UML diagrams; and the Standard Widget Toolkit [7], which is used to provide additional graphical user interface where needed. The SMT is continuously under development.

The rest of this paper is structured as follows: Section 2 provides an overview on the methodology, Section 3 presents and illustrates the use of the Security Modeling Tool, Section 4 elaborates on the SMT architecture and design, and Section 5 summarizes and proposes ideas for future work.

2 Methodology Overview

The methodology can be roughly divided into four phases: preparation, analysis, design, and implementation. **Fig. 1** presents the scope of the methodology in terms of the tasks to be performed in each phase (presented in round rectangle) and the generated artifacts (presented in rectangle). The preparation phase occurs at the organizational level, whereas the other three phases occur at the application development level.

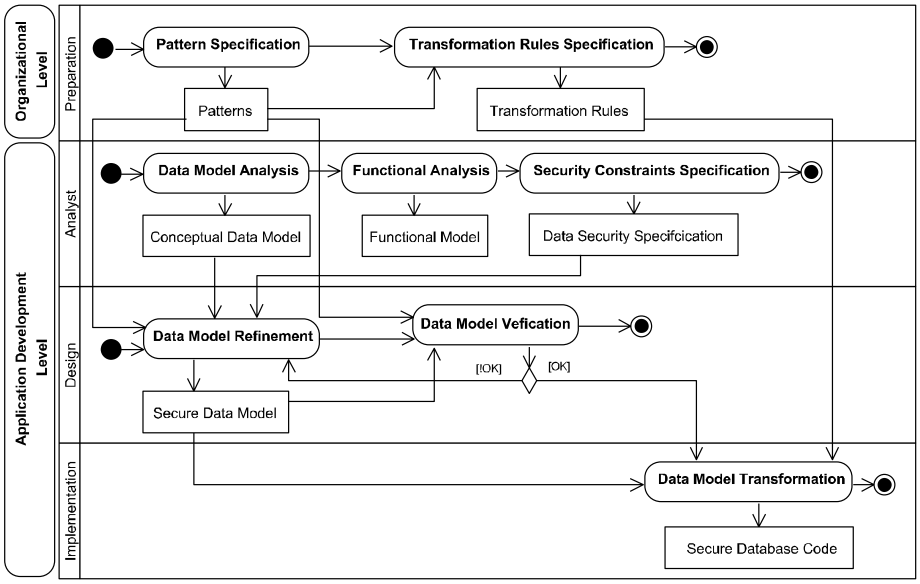


Fig. 1. Methodology overview

At the organizational level, in which the **preparation phase** takes place, we define organizational security policies in the form of security patterns. These security patterns present general access control policies within the organization. Once the patterns are specified, the transformation rules are defined, depicting how to transform a logical model, based on the pattern, into a database code. The artifacts created in this phase are reusable and may be applied to various applications.

The application level deals with the development of different applications within the organization. In the **analysis phase** of the application development process, two models are defined, following the FOOM methodology [6]: a conceptual data model in the form of an initial class diagram, and a functional model in the form of extended use cases. Then, the security constraints regarding authorization to access the database are analyzed and specified in natural language. In the **design phase**, the artifacts from the preparation and analysis stage are used to refine the data model and enhance it with the definitions of the security patterns, in order to create a secure data model. Next, the secure data model is verified. If the verification fails, the data model is refined until it adheres to the rules of the security patterns. In the **implementation phase**, the secure data model is transformed into a secure database schema with its access control specifications. This process is performed by executing the transformation rules specified in the preparation phase as part of the security patterns.

3 The Security Modeling Tool

3.1 Organizational Level - The Preparation Phase

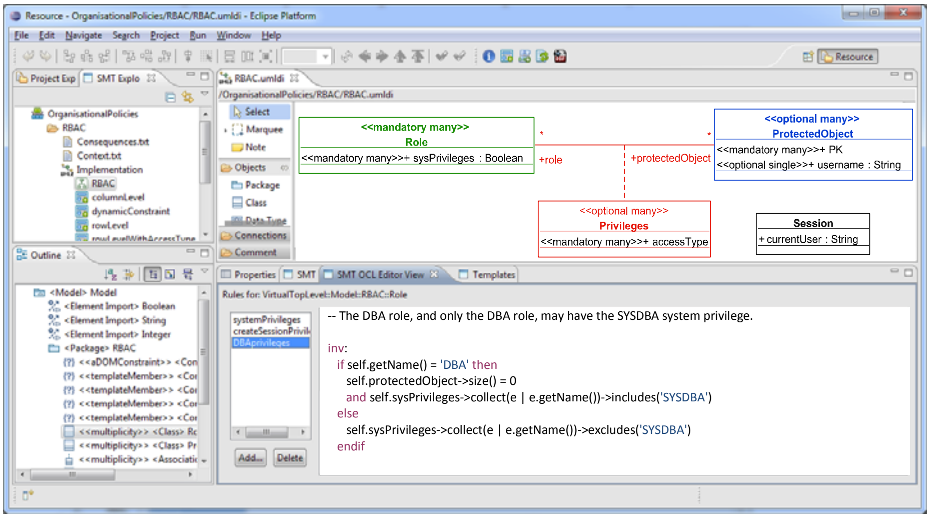
During the preparation phase, security patterns along with their transformation rules are specified. These patterns will serve as guidelines for application developers as well as a verification template. In addition, they provide the infrastructure for the transformation process.

Security Pattern Specification: Similarly to the classical pattern approach, security patterns are specified in a structured form. The standard template aids designers, who are not security experts, to identify and understand security problems and solve them efficiently. In order to specify the patterns, we use a common template introduced by Schumacher [5]. The template consists of five main sections: *name, context, problem, solution,* and *consequence*. The *name*, *context*, *problem*, and *consequence* sections are documentation text files; the SMT provides a text editor to support the specification of these sections. They provide the *name* of the pattern, the *context* in which the security problem occurs, the description of the security *problem*, and the *consequences* of this solution. The *solution* section provides a generic solution to the problem. It is specified with a UML class diagram that provides the static structure of the solution. The SMT uses a UML editor that is based on TOPCASED [9]. Fig. 2 (upper side) presents the structure of a simple Role-Based Access Control (RBAC) pattern. In the described pattern, *Role* is akin to an external group of entities or users playing a specific role that needs to access the database. While applying or implementing this RBAC pattern, it is obligatory to define at least one *Role* as it is defined as a <<*mandatory*>> element. In addition, one can specify the system privileges assigned to some *Role* by using the *sysPrivileges* classification *ProtectedObject* is akin to a database table, where the *PK* classification is used to indicate the primary keys of the table. *Privileges* association class determines the schema object privileges of a *Role* with respect to a specific *ProtectedObject*. A class that is classified as *Privileges* must include at least one object privilege – *accessType*. Both *sysPrivileges* and *accessType* classifications are Boolean properties that should be assigned to TRUE in case a privilege is given.

In addition, OCL constraints are used to specify additional constraints that cannot be expressed via the diagrams. The SMT provides an OCL Editor to add constraints to the pattern. These OCL constraints are evaluated in the application layer during the verification in the design stage, rather than in the domain layer where they are defined. To enable this verification we had to define several operators, such as *getName()* or *getParent()*, that support metadata queries on the elements.

The lower part of Fig. 2 shows an example of an OCL rule. In this example, the OCL rule restricts the number of roles that can have the SYSDBA system privilege to one, and that is the DBA role. Another example of such OCL rule is the following constraint that limits object privileges to `SELECT`, `INSERT`, `UPDATE` and `DELETE`:

```
context Privileges
inv: Set{'SELECT','INSERT','UPDATE','DELETE'}->
        includesAll(self.accessType->collect(e|e.getName()))
```



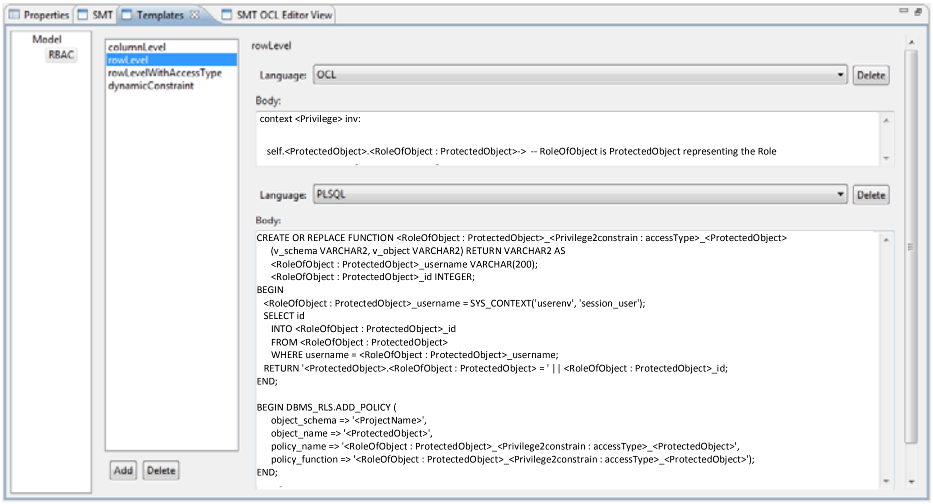
Rules for: VirtualTopLevel::Model::RBAC::Role

```
-- The DBA role, and only the DBA role, may have the SYSDBA system privilege.

inv:
  if self.getName() = 'DBA' then
    self.protectedObject->size() = 0
    and self.sysPrivileges->collect(e | e.getName())->includes('SYSDBA')
  else
    self.sysPrivileges->collect(e | e.getName())->excludes('SYSDBA')
  endif
```

Fig. 2. Pattern specification window

In case that a finer grained solution is required, SMT provides a method to define OCL rules in the form of general templates [8]. These general templates are specified using the specific elements that were already defined by the class diagrams specifying the structure of the pattern. In the RBAC example, the *Role*, *ProtectedObject*, *accessType* are some of those elements. The templates are essentially exemplars of the desired output code with "blanks" that should be filled in with a value of an attribute. These "blanks" contain meta-code and are delimited between "< >". After the missing values are inserted, a template engine is used to create the output code. Fig. 3 presents the *instance level* template that is used to specify access constraints on an instance of an object (or a row of a table in terms of relational database). These templates are used to specify fine grained access control policies during the application modeling. The developers need only to fill in the missing parameters that are inside the triangle brackets and do not need to write code in PL/SQL unless they want to express some complex constraint.

Once the *solution* (i.e., the pattern) is defined, the SMT automatically generates a UML profile, which will be used by the applications to classify security elements. In our case the profile will consist of the following: the stereotypes *Role* and *ProtectedObject* are created and are associated with the class meta element, the *Privilege* stereotype is associated with the association class meta element. The attributes of *sysPrivileges*, *accessType, PK,* and *username,* are created as stereotypes associated with the property meta element. The various OCL constraints are also transformed into the profile. Note that we did not associate any new notations for the profiles; rather we used the standard <<stereotype>> to add semantics to the application model elements.



```
context <Privilege> inv:


  self.<ProtectedObject>.<RoleOfObject : ProtectedObject>->  -- RoleOfObject is ProtectedObject representing the Role
```

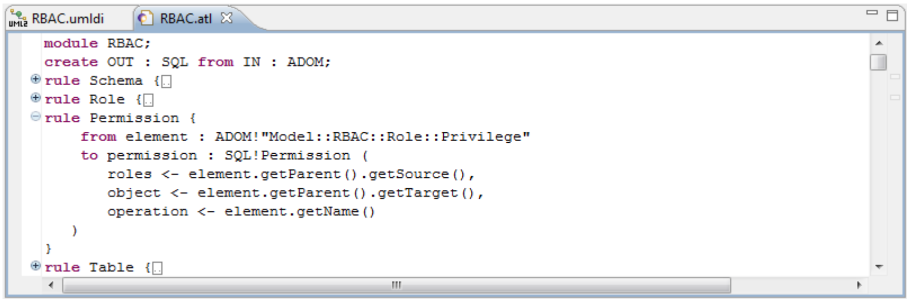
```
CREATE OR REPLACE FUNCTION <RoleOfObject : ProtectedObject>_<Privilege2constrain : accessType>_<ProtectedObject>
    (v_schema VARCHAR2, v_object VARCHAR2) RETURN VARCHAR2 AS
    <RoleOfObject : ProtectedObject>_username VARCHAR(200);
    <RoleOfObject : ProtectedObject>_id INTEGER;
BEGIN
  <RoleOfObject : ProtectedObject>_username = SYS_CONTEXT('userenv', 'session_user');
  SELECT id
    INTO <RoleOfObject : ProtectedObject>_id
    FROM <RoleOfObject : ProtectedObject>
    WHERE username = <RoleOfObject : ProtectedObject>_username;
  RETURN '<ProtectedObject>.<RoleOfObject : ProtectedObject> = ' || <RoleOfObject : ProtectedObject>_id;
END;

BEGIN DBMS_RLS.ADD_POLICY (
    object_schema => '<ProjectName>',
    object_name => '<ProtectedObject>',
    policy_name => '<RoleOfObject : ProtectedObject>_<Privilege2constrain : accessType>_<ProtectedObject>',
    policy_function => '<RoleOfObject : ProtectedObject>_<Privilege2constrain : accessType>_<ProtectedObject>');
END;
```

Fig. 3. The Instance (Row) Level Template

Transformation Rules Specification: To transform an application model (UML class diagram) into SQL code, we use the ATLAS Transformation Language (ATL) [3]. These transformation rules are generic and refer to all applications. The ATL rules specify how the application elements should be transformed into SQL elements. So, applications are transformed to SQL model instantiating an SQL meta-model provided by SMT. Then, the SQL model (created by the ATL transformation) is automatically converted to SQL code by the SMT. Fig. 4 shows the transformation rule for *Privilege*.

3.2 Application Development Level

To demonstrate the use of SMT at the application development level, we use a simple university system, which enables to register students to courses, update student details, assign grades, etc. Naturally, each system operator has different privileges.



```
module RBAC;
create OUT : SQL from IN : ADOM;
rule Schema {
rule Role {
rule Permission {
    from element : ADOM!"Model::RBAC::Role::Privilege"
    to permission : SQL!Permission (
       roles <- element.getParent().getSource(),
       object <- element.getParent().getTarget(),
       operation <- element.getName()
    )
}
rule Table {
```

Fig. 4. ATL transformation code for the Privilege association class

The Analysis Phase: The first task in the analysis phase is to create a conceptual data model from the users' requirements. The conceptual data model is an initial class diagram that consists of data classes, their attributes and various types of relationships. Fig. 5 depicts the initial (UML) class diagram of a university registration system.

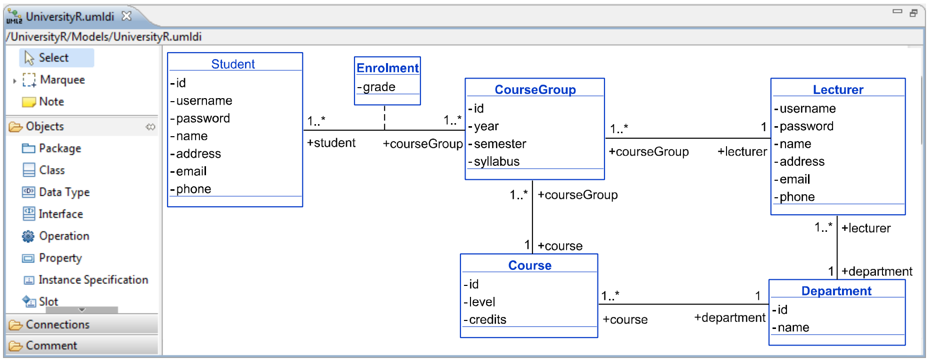


Fig. 5. An example of an initial class diagram

Next, the functional model of the application is defined using **extended use cases** (EUC). A EUC is similar to a FOOM transaction [6]; it includes, besides the functions of the UC, also external/user entities and data classes. An external/user entity provides input data or obtains output information from the system. (It is different from an Actor in ordinary use cases, which only signify who operates the use case.) Data classes, which are taken from the initial class diagram, are manipulated (i.e., retrieved or updated) by the functions of the EUC. As in ordinary use-cases, for every EUC diagram we also prepare a description. The template for a EUC description is extended compared to an ordinary UC description, as it includes definitions of access privileges.

Later on in the development process, for each class included in a EUC the developer defines: a) the authorized operators (i.e., roles) of the EUC; b) the type of access privilege (e.g., add, read, update or delete); and c) the attributes involved in that operation. Fig. 6 shows an example of a EUC diagram that is supported by the EUC editor. A *Student* is an external entity which provides inputs and gets outputs, the *Course*, *Course Offering* and *Enrollment* are classes, and *Display courses*, *Display selected courses' offerings*, and *Add registration to selected course* are functions. The EUC editor extends the TOPCASE use case diagram notations to support the new elements, i.e., classes and the different types of links.

Fig. **7** shows part of the EUC description that is supported by the EUC Analysis Editor. At the bottom of Fig. **7**, the *security specifications* section is presented in a form of a table, where for each class that participates in the EUC the access control privileges are specified. Eventually, all the security specifications, defined for all the EUCs are aggregated in one table.

It should be noted that EUC diagrams also serve as the core functional/behavioral model of the application, and can be used for the generation of the input and output forms and reports, as well as skeleton of the code.

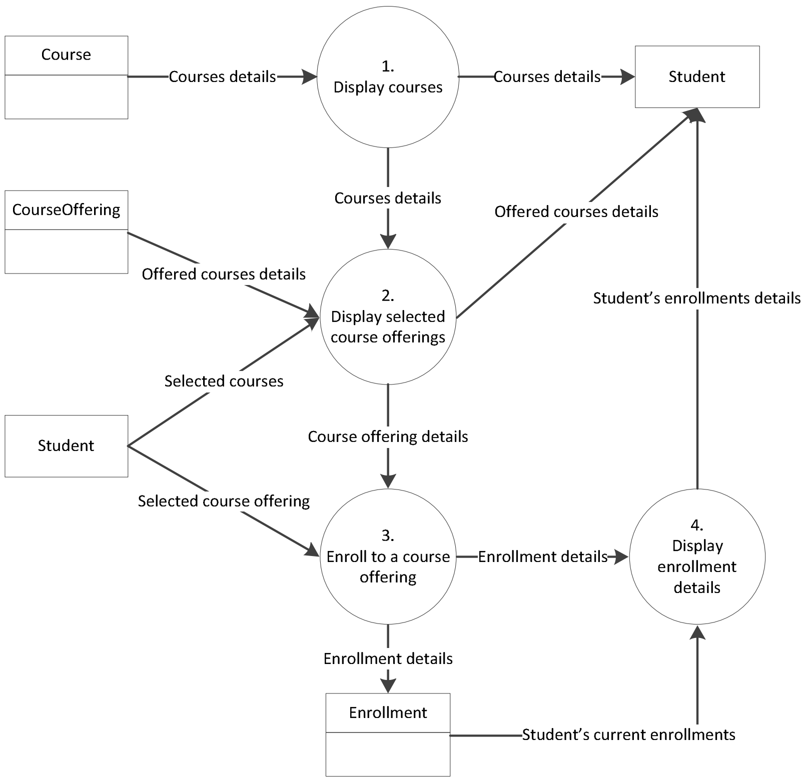
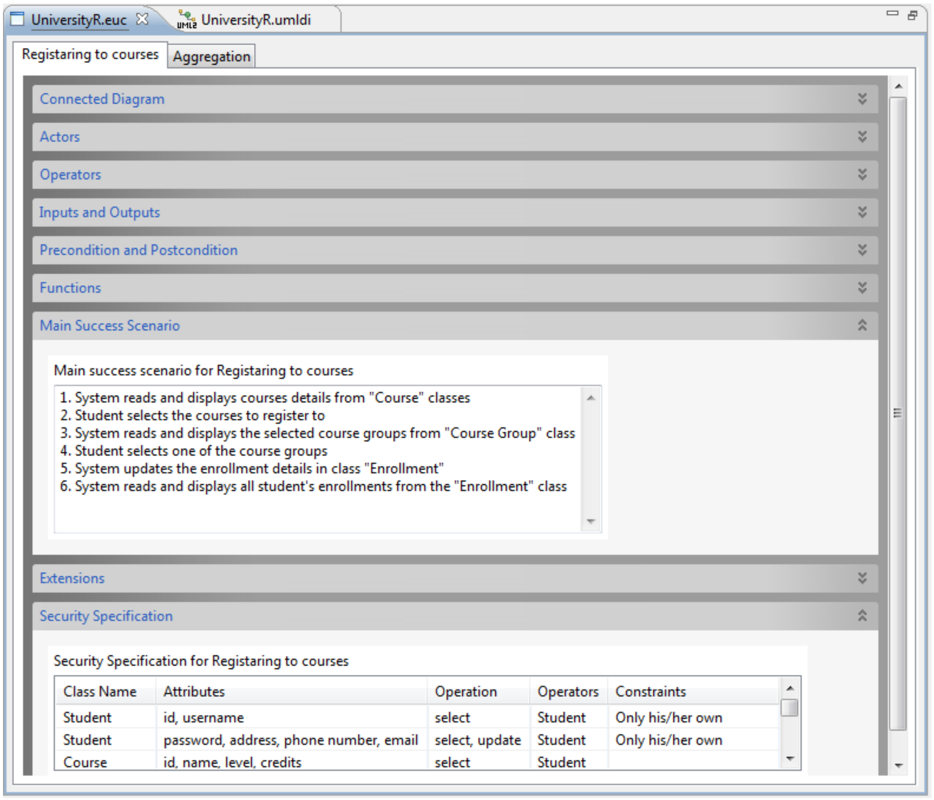


Fig. 6. An example of a EUC diagram

The Design Phase: During this phase, the initial class diagram is refined by the designer, to include the security specification. The SMT allows the designer to specify which security patterns are used in the application. Then, the various elements that appear in the initial class diagram are classified according to the security patterns defined in the preparation stage. Technically, this is done by assigning the stereotypes from the security pattern profile that was created when the security pattern was finalized. Then, the SMT allows the designer to select stereotypes for each element according to the applied patterns, and the element type. .**Fig. 8** presents the refined data model of the university application. In that figure, the relevant classes are associated with the *Role* and *ProtectedObject* stereotypes and new *Privileges* classes are introduced. These include the names of the *accessType* attributes as set by the OCL constraint, and the initial values of these attributes (which are not shown visually, yet they are part of the model); in the example their values are True.



Main success scenario for Registaring to courses

1. System reads and displays courses details from "Course" classes
2. Student selects the courses to register to
3. System reads and displays the selected course groups from "Course Group" class
4. Student selects one of the course groups
5. System updates the enrollment details in class "Enrollment"
6. System reads and displays all student's enrollments from the "Enrollment" class

Security Specification for Registaring to courses

Class Name	Attributes	Operation	Operators	Constraints
Student	id, username	select	Student	Only his/her own
Student	password, address, phone number, email	select, update	Student	Only his/her own
Course	id, name, level, credits	select	Student	

Fig. 7. An example of a EUC description

During the design phase, additional changes to authorization rules may be applied and fine grained restrictions may be specified using the templates that were defined in the patterns. The templates are instantiated using the Template Editor. Fig. 9 illustrates the use on the instance (row) level template that was defined in Fig. 3. To use the template, the designer merely instantiates it and provides the missing parameters. The Template Editor lists the missing parameters at the bottom. The SMT also provides a preview of the templates after the missing parameters were specified.

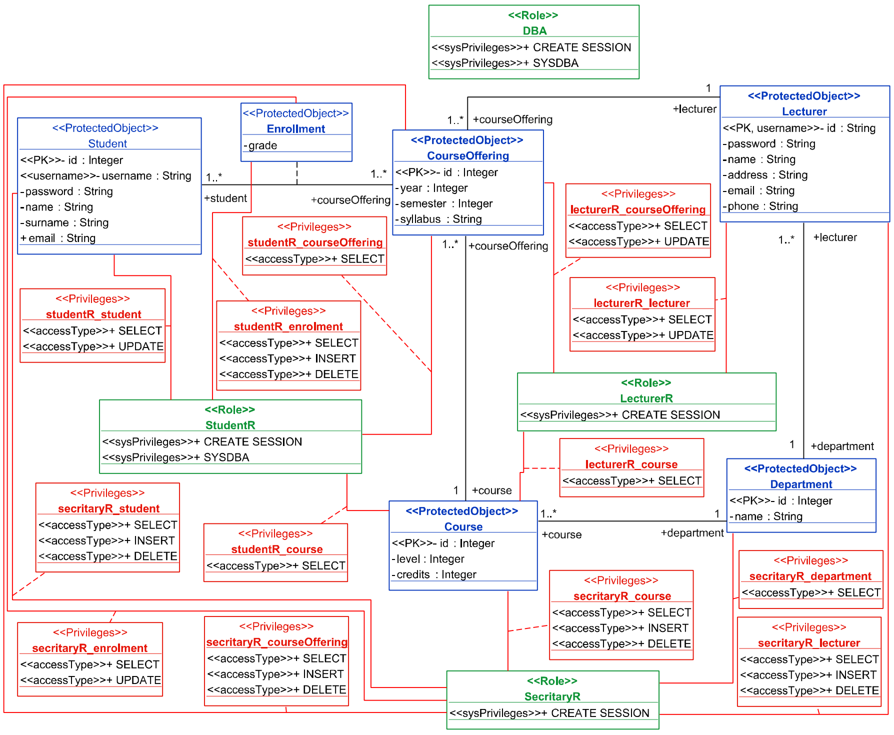
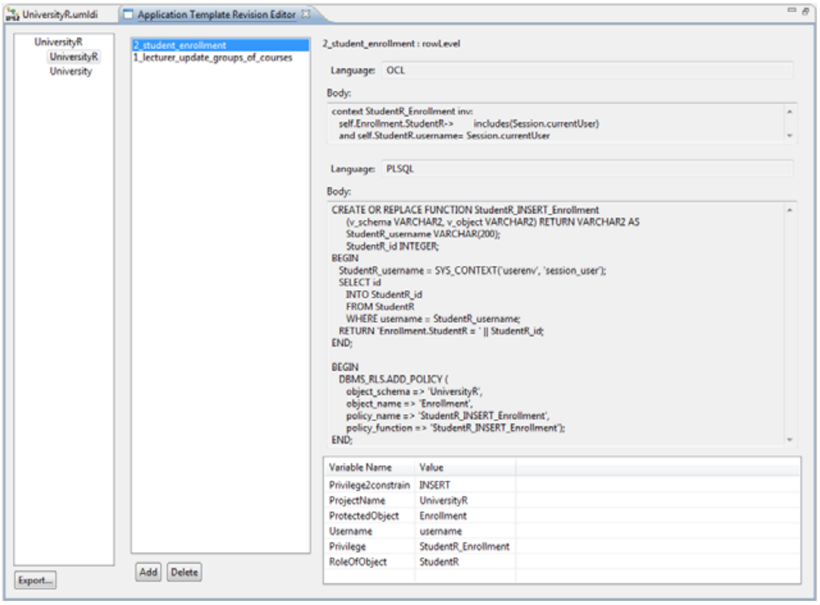


Fig. 8. An example of the RBAC-base refined data model



2_student_enrollment : rowLevel

Language: OCL

Body:

```
context StudentR_Enrollment inv:
  self.Enrollment.StudentR->      includes(Session.currentUser)
  and self.StudentR.username= Session.currentUser
```

Language: PLSQL

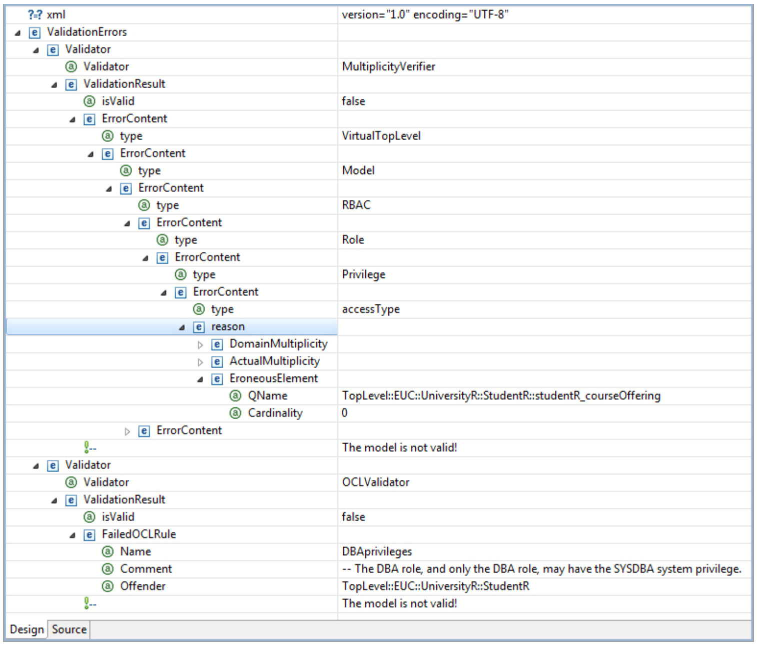
Body:

```
CREATE OR REPLACE FUNCTION StudentR_INSERT_Enrollment
    (v_schema VARCHAR2, v_object VARCHAR2) RETURN VARCHAR2 AS
    StudentR_username VARCHAR(200);
    StudentR_id INTEGER;
BEGIN
  StudentR_username = SYS_CONTEXT('userenv', 'session_user');
  SELECT id
    INTO StudentR_id
    FROM StudentR
    WHERE username = StudentR_username;
  RETURN 'Enrollment.StudentR = ' || StudentR_id;
END;

BEGIN
  DBMS_RLS.ADD_POLICY (
    object_schema => 'UniversityR',
    object_name => 'Enrollment',
    policy_name => 'StudentR_INSERT_Enrollment',
    policy_function => 'StudentR_INSERT_Enrollment');
END;
```

Variable Name	Value
Privilege2constrain	INSERT
ProjectName	UniversityR
ProtectedObject	Enrollment
Username	username
Privilege	StudentR_Enrollment
RoleOfObject	StudentR

Fig. 9. An example of instance level (row) constraint in OCL and PL/SQL

After creating a refined data model, we need to check if it adheres to the security policies as defined by the specified security patterns. The SMT provides automatic verification. This verification is essentially a conformance checking with respect to the relevant patterns; it includes checking the number of elements, as depicted in [4], their types, and the available OCL constraints. If the application is invalid, an error message, like the one appears in Fig. 10, is presented, explaining the verification errors. In that example there are two errors: 1) multiplicity error: access type is not specified to the *Privilege* class *StudentR_CourseOffering*; 2) OCL error: *StudentR* role has the *SYSDBA* privilege.



```
?-? xml                                   version="1.0" encoding="UTF-8"
◢ e ValidationErrors
  ◢ e Validator
      a Validator                         MultiplicityVerifier
    ◢ e ValidationResult
        a isValid                         false
      ◢ e ErrorContent
          a type                          VirtualTopLevel
        ◢ e ErrorContent
            a type                        Model
          ◢ e ErrorContent
              a type                      RBAC
            ◢ e ErrorContent
                a type                    Role
              ◢ e ErrorContent
                  a type                  Privilege
                ◢ e ErrorContent
                    a type                accessType
                  ◢ e reason
                    ▷ e DomainMultiplicity
                    ▷ e ActualMultiplicity
                    ◢ e EroneousElement
                        a QName           TopLevel::EUC::UniversityR::StudentR::studentR_courseOffering
                        a Cardinality     0
            ▷ e ErrorContent
        --                                The model is not valid!
  ◢ e Validator
      a Validator                         OCLValidator
    ◢ e ValidationResult
        a isValid                         false
      ◢ e FailedOCLRule
          a Name                          DBAprivileges
          a Comment                       -- The DBA role, and only the DBA role, may have the SYSDBA system privilege.
          a Offender                      TopLevel::EUC::UniversityR::StudentR
        --                                The model is not valid!
Design | Source
```

Fig. 10. An example of an error massage

The Implementation Phase: During this phase the transformation rules, which were defined during the preparation phase, are used to translate the verified application model into database code. Fig.11 presents the generated SQL commands for the *Student* role and a sample of the SQL fine-grained code for the university application.

The artifacts produced in the organizational level, and the artifacts leading to the implementation in the application development level, can be exported as documentation in a PDF file.

More details on the pattern-based approach that is applied as part of the methodology can be found in [14].

```
-- Role creation
CREATE ROLE STUDENT;
-- Granting privileges to Student
GRANT CREATE SESSION TO STUDENT;
GRANT SELECT ON COURSE_OFFERING TO STUDENT;
GRANT SELECT ON COURSE TO STUDENT;
GRANT SELECT, INSERT, DELETE ON ENROLLMENT TO STUDENT;
GRANT SELECT, UPDATE ON STUDENT TO STUDENT;
-- Instance level template transformation
-- Students can update only their personal information:
CREATE FUNCTION STUDENT_STUDENT_UPDATE
   (SCHEMAV VARCHAR2, OBJ VARCHAR2) RETURN VARCHAR2 AS
BEGIN
  IF (NOT DBMS_SESSION.IS_ROLE_ENABLED('STUDENT')) THEN
    RETURN NULL;
  END IF;
  RETURN 'username = ' || SYS_CONTEXT('USERENV', 'SESSION_USER');
END;
BEGIN DBMS_RLS.add_policy(
  object_schema   => 'UNIVERSITY',
  object_name     => 'STUDENT',
  policy_name     => 'STUDENT_STUDENT_UPDATE ',
  policy_function => 'STUDENT_STUDENT_UPDATE ',
  statement_types => 'UPDATE',
  update_check    => TRUE);
END;
```

Fig. 11. A sample of the generated SQL commands for the university application

4 SMT Architecture and Design

In this section, we discuss the implementation details of SMT. We first introduce the technologies on which SMT is based, as well as the reasons for choosing them. Then, we elaborate on the specific components developed within SMT, i.e., the different editors and the ADOM library. Finally, we describe how the SMT components interact with each other. Fig. 12 shows the components that SMT uses (marked in a broken line), and the components that were developed internally (marked in a solid line). The figure also shows the dependencies among the various components, which are organized as layer of dependencies. Note that the integration of all these components is done using the plug-ins facilities of the Eclipse framework. In short, at the **organization level**, in order to specify the pattern, SMT uses *text editors* for the pattern description, *UML editor* for the specification of the structure of the pattern along with the ADOM library, *OCL editor* to specify the constraints on the structure of the pattern, and *Templates editor* to specify the fine grain templates. Then, for specifying the pattern transformation we use ATL. At the **application development level**, the *UML and EUC editors* are used to specify the different application diagrams, *EUC analysis editor* to define the textual description of the EUC, the *ADOM library* to refine and verify the application data model by the pattern, the *Dresden OCL* to verify that the OCL constraints hold, and finally, ATL to transform the application model into code.

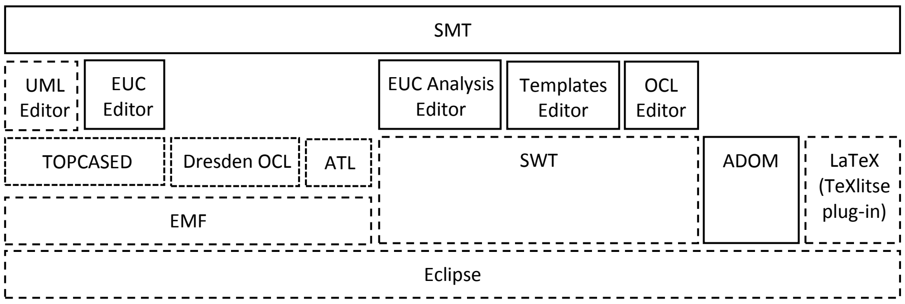


Fig. 12. Overview of the SMT components

4.1 Eclipse

SMT and all its dependencies rely on the framework provided by Eclipse [10]. It was chosen for the following reasons: (a) it is an extensible platform; (b) it is open source, meaning that the source code can be used as documentation and aid in finding errors; and (c) the Eclipse framework has been extended to support many technologies, including UML by TOPCASED [9] and OCL [11].

4.2 EMF

Much of the SMT's functionality relies on modeling capabilities. This functionality is provided by the Eclipse Modeling Framework (EMF) project [2], which is a modeling framework and code generation facility for building tools and other applications based on a structured data model.

EMF is a very mature modeling framework, providing a complete toolset for working in a model-oriented context. Additionally, it is available as an Eclipse extension, making it highly suitable for our needs. Many model-oriented features in Eclipse, and all such features used by SMT, rely on EMF. Note also that Eclipse's UML library too is based on EMF.

4.3 TOPCASED UML Editor

TOPCASED [9] is an Eclipse plug-in that provides a simple, extendible graphical modeling framework. TOPCASED also provides a UML editor as one of its sub-projects. Additionally, this framework may be extended to provide graphical editors to other diagram types. This capability was used within the general UML editors and to create and provide a new editor for the EUC diagrams. TOPCASED relies on EMF for both persistency and representation. Therefore, other modules based on EMF may interact with the artifacts generated by TOPCASED without the need for a new API or adaptor.

4.4 Dresden OCL

Dresden OCL [11] provides a set of tools to parse and evaluate OCL constraints on various models like UML, EMF, and Java. Furthermore, it provides tools for Java/AspectJ and SQL code generation. The tools of Dresden OCL can be either used as a library for other project or as a plug-in project that extends Eclipse with OCL support. In the case of SMT, the Dresden OCL library is used to enforce OCL constraints. As EMF provides only syntactical solutions, complex semantic solutions are beyond the scope of the EMF project. In SMT, we found that OCL may be specified using EMF's implementation of OCL; however EMF was not flexible enough to allow us to interpret and enforce the OCL rules on its own, and so the Dresden OCL library was used.

4.5 ATL

ATLAS Transformation Language (ATL) [12] is a model transformation language and toolkit. In the field of Model-Driven Engineering (MDE), ATL provides ways to produce a set of target models from a set of source models. SMT uses ATL to define a transformation from a pattern-based application design to its equivalent in SQL.

4.6 SWT

The Standard Widget Toolkit (SWT) [13] is an open source, graphical widget toolkit used to develop user interfaces in Java. It is a pre-built part of the Eclipse framework, which allows user interfaces implemented in SWT to be integrated more natively into Eclipse than other Java GUI toolkit. For this reason, user interfaces designed for SMT were implemented using SWT rather than a different widget framework.

4.7 The ADOM Library

The relationship between the organizational security pattern models and the application models within SMT are provided by the ADOM library. As we plan to adopt ADOM in various modeling notations (e.g., class and sequence diagrams), it was designed to be language independent. The ADOM library is separated into two main categories to allow for implementation in various languages and environments: language independent code, and language dependent code. That library also implements the ADOM validation algorithm.

Language independent code makes no assumptions on the used language beyond what is defined by ADOM. That is the multiplicity indicator which is defined as a UML profile. That part of the tool has three sections which are relevant to SMT:

a) The **abstract data-structure,** which provides a tree-like structure of model elements;
b) The **element options**, which allow ADOM elements to be extended to include data and functionality deemed necessary by the developer of the library’s extension. For instance, the multiplicity validation algorithm extends each ADOM element to contain required and actual multiplicity. The OCL validation algorithm extends each ADOM element to contain any number of OCL constraints, which must be confirmed in order for the validation to succeed.
c) The **validation algorithms.** This is also a pluggable mechanism, used to allow library extension developers to provide validation algorithms on ADOM applications in reference to their ADOM domains. Usually, validation algorithms also provide element options, providing them with additional data necessary to perform the validation algorithm, such as multiplicity and OCL constraints as stated before. Note that these parts were reused in other ADOM-related projects.

The language dependent code has two sections relevant to SMT. The first section is an implementation of the abstract data-structure described in the language independent part. This implementation may be used by other components to have direct access to the underlying data-structures of the language in use, and retrieve necessary data.

4.8 Document Generation

The document generation facilities of SMT are provided using LaTeX. LaTeX is a high-quality typesetting system that includes features designed for the production of technical and scientific documentation. SMT generates a LaTeX document, which is then processed and generates an output file in PDF. The SMT enables the document generation of both organizational policies and application specification.

5 Summary

We have presented SMT, a Security Modeling Tool, which supports the development of secured database schemata following a methodology that we have developed. This tool utilizes security patterns for guiding and enforcing security on database application design. The tool guides developers on how to incorporate security aspects defined by security patterns, in particular authorization, within the development process. It handles the specification and implementation of the authorization aspect from the early stages of the development process, leading to a secure system design.

In this paper we demonstrated the application of an access control policy (RBAC) over a database. We also implemented other policies such as DAC and MAC using the same methodology. In addition, we are in the process of using the same mechanisms to implement patterns other than access control to other software layers besides the database. Currently, we are in a process of applying the methodology along with its supporting tool in an industrial environment. This will enable us to introduce improvements in the methodology and the tool. In future work, we plan to enrich the methodology and tool to support other security requirements (e.g., privacy, encryption, and auditing). In addition, we plan to further extend the methodology to deal also with the behavioral specification of applications, in addition to its application in structural specification.

References

1. Dhillon, G.S.: Information Security Management: Global Challenges in the New Millennium. IGI Publishing (2001)
2. Eclipse Modeling Framework (2011), `http://www.eclipse.org/modeling/emf/`
3. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: A model transformation tool. Science of Computer Programming. Science of Computer Programming 72(1-2), 31–39 (2008)
4. Reinhartz-Berger, I., Sturm, A.: Utilizing Domain Models for Application Design and Validation. Information & Software Technology 51(8), 1275–1289 (2009)

5. Schumacher, M.: Security Engineering with Patterns: Origins, Theoretical Models, and New Applications. Springer-Verlag New York, Inc., Secaucus (2003)
6. Shoval, P.: Functional and Object-Oriented Analysis and Design - An Integrated Methodology. IGI Publishing, Hershey (2007)
7. Standard Widget Toolkit (2011), http://www.eclipse.org/swt/
8. StringTemplate (2011), http://www.stringtemplate.org/
9. TOPCASED (2011), http://www.topcased.org/
10. Eclipse (2011), http://www.eclipse.org/
11. Dresden OCL Toolkit (2011), http://www.dresden-ocl.org/index.php/DresdenOCL
12. ATL (2011), http://eclipse.org/atl/
13. Standard Widget Toolkit (2011), http://www.eclipse.org/swt/
14. Abramov, J., Sturm, A., Shoval, P.: A Pattern Based Approach for Secure Database Design. In: Salinesi, C., Pastor, O. (eds.) CAiSE Workshops 2011. LNBIP, vol. 83, pp. 637–651. Springer, Heidelberg (2011)
15. Basin, D., Doser, J., Lodderstedt, T.: Model driven security: From UML models to access control infrastructures. ACM Transaction on Software Engineering and Methodologies 15(1), 39–91 (2006)
16. Fernandez, E.B., Larrondo-Petrie, M.M., Sorgente, T., VanHilst, M.: A methodology to develop secure systems using patterns. In: Mouratidis, H., Giorgini, P. (eds.) Integrating Security and Software Engineering: Advances and Future Vision. IDEA Press (2006)
17. Lodderstedt, T., Basin, D., Doser, J.: SecureUML: A UML-Based Modeling Language for Model-Driven Security. In: Jézéquel, J.-M., Hussmann, H., Cook, S. (eds.) UML 2002. LNCS, vol. 2460, pp. 426–441. Springer, Heidelberg (2002)
18. Mouratidis, H., Giorgini, P.: Secure Tropos: a Security-Oriented Extension of the Tropos Methodology. International Journal of Software Engineering and Knowledge Engineering 17, 285–309 (2007)
19. Jurjens, J.: Secure Systems Development with UML. Springer (2005)
20. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. Addison-Wesley Professional (1995)
21. Hafner, M., Breu, R.: Security Engineering for Service oriented Architectures. Springer (2009)
22. Ray, I., France, R.B., Li, N., Georg, G.: An aspect-based approach to modeling access control concerns. Information & Software Technology 46, 575–587 (2004)
23. Fernández-Medina, E., Piattini, M.: Designing secure databases. Information & Software Technology 47(7), 463–477 (2005)
24. Mouratidis, H., Jurjens, J.: From goal-driven security requirements engineering to secure design. International Journal on Intelligent Systems 25(8), 813–840 (2010)
25. Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., Sommerlad, P.: Security Patterns: Integrating Security and Systems Engineering. John Wiley & Sons (2006)

Towards Agile Model-Driven Web Engineering[*]

José Matías Rivero[1,2], Julián Grigera[1], Gustavo Rossi[1,2], Esteban Robles Luna[1,3], and Nora Koch[4,5]

[1] LIFIA, Facultad de Informática, UNLP, La Plata, Argentina
{mrivero,julian.grigera,gustavo, esteban.robles}@lifia.info.unlp.edu.ar
[2] Also at Conicet
[3] Also at CIC
[4] Ludwig-Maximilians-Universität München
[5] Cirquent GmbH, Germany
kochn@pst.ifi.lmu.de

Abstract. The increasing growth of the Web field has promoted the development of a plethora of Model-Driven Web Engineering (MDWE) approaches. These methodologies share a top-down approach: they start by modeling application content, then they define a navigational schema, and finally refine the latter to obtain presentation and rich behavior specifications. Such approach makes it difficult to acquire quick feedback from customers. Conversely, agile methods follow a non-structured, implementation-centered process building software prototypes to get immediate feedback. In this work we propose an agile approach to MDWE methodologies (called Mockup-Driven Development, or MockupDD) by inverting the development process: we start from user interface mockups that facilitate the generation of software prototypes and models, then we enrich them and apply heuristics in order to obtain software specifications at different abstraction levels. As a result, we get an agile prototype-based iterative process, with advantages of a MDWE one.

Keywords: Mockups, User Interface, Agile, Web Engineering, MDD.

1 Introduction

During the last 20 years, many Model-Driven Web Engineering (MDWE) methodologies have been defined to improve the development process of web applications approaches [1-4]. These methodologies share a common top-down approach [5] and construct web applications by describing a set of models at different levels of abstraction:

- *Content (or Domain) Model*: defining domain objects and their relationships.
- *Hypertext (or Navigation) Model*: defining navigation nodes and links that publish and manipulate information specified by objects in the *Content Model*.

[*] This work is an extended version of the paper "Improving Agility in Model-Driven Web Engineering", published in CEUR, Vol. 734.

- *Presentation Model*: refining the *Hypertext Model* with concrete user interface presentation features like pages, concrete widgets, layout, etc.

This process is generally top-down, delivering a final web application through a process of (sometimes automatic) model transformations which maps the previously described models into other models or a specific technology.

Agile methodologies, on the other hand, promote early and constant interaction with customers to assert that the software built complies with their requirements, by constantly delivering prototypes developed in short periods of time; application prototypes are then used as some kind of *common language* between developers and final users to assert captured requirements and to discover new ones. Agile approaches argue that software specifications must emerge naturally, enhancing former prototypes along the development until the final application is obtained.

To summarize, while MDWE methodologies facilitate software specification portability, abstraction and productivity, they fail in providing *agile* interaction with customers because concrete results are obtained too late. On the other hand, while this feature is clearly provided by agile methodologies, they are heavily based on direct implementation and thus fail to provide abstraction, portability and productivity through automatic code-generation.

In this paper we propose an hybrid model-based agile methodology – called Mockup-Driven Development (MockupDD) – aiming to extract the best of both worlds, i.e. a process driven by the active participation of users and customers, and a classical approach following the phases of analysis, design and implementation assisted with the use of models in all stages. Our approach starts by the requirement analysis, i.e. defining mockups (ideally together with the customers) to agree upon the application's functionality, similar to Harel's behavioral programming approach [6]. Then, mockups are translated to an abstract user interface model that can be directly derived to specific MDWE presentation models or technology-dependent UI prototypes. By tagging mockups and presentation models we add navigation features, and based on the navigation specification, we use heuristics to infer content models. Thus, we are starting the requirement specifications with objects that are perceivable by customers (UI structure elements), easing requirements gathering and traceability [7].

Therefore, since we start with presentation models obtained from mockups and then construct or obtain *upper* (i.e. abstract) models, we are inverting the traditional MDWE process, yielding to a more *agile*, yet truly model-based approach. While we exemplify with the UML-based Web Engineering (UWE) [3], MockupDD can be applied to any MDWE approach.

2 Related Work

User Interface (UI) Mockup tools like Balsamiq[1], Pencil[2] or Mockingbird[3] suit well in agile methodologies [8-10], since they provide a quick and easy way of capturing

[1] Balsamiq - http://balsamiq.com/, last visited 3/9/2011.

[2] Pencil Project - http://pencil.evolus.vn/en-US/Home.aspx, last visited 3/9/2011.

[3] Websites wireframing: Mockingbird - https://gomockingbird.com, last visited 3/9/2011.

interaction requirements. Usually, mockups are defined in companion with other specifications like use cases [11, 12], user stories [13] or informal annotations [14]. Also, mockups have been introduced in the context of model-driven development (MDD) approaches as can be appreciated in the use of UI sketches in the context of the ConcurTaskTrees [15] to express interaction requirements and in the definition of a language that introduces storyboards over user interfaces composed through a specific user interface widget set [16]. Finally, the result of statistical studies [17] conducted over inexperienced software engineers asserted that mockups effectively increases and eases software comprehension.

In most cases, however, mockups themselves are not considered as models and they are usually thrown away after requirement modeling. Thus, mockups are not used as important drivers of the development process although they contain precise information about the users' needs. In our previous work [18], we introduced the idea of translating mockups constructed with prototyping tools like Balsamiq to a common presentation language in order to preserve and reuse them as truly software specifications. While this approach facilitates both a "*quick and dirty*" way of user interface construction with intense customer participation and a fast method to generate high quality UI from models (something that is not currently supported by well known MDWE approaches), it lacks the capacity of defining non-presentational features. As a consequence, mockups (and final UIs generated from them) can be used as a common language to gather further non-presentational requirements but these requirements must be coded by hand, missing the agility provided by automatic code generation in the context of a model-driven process. Here we go a step further and propose not only mockup reusing but the specification of advanced features like navigation and content through the application of a set of lightweight enrichments directly over them. This makes our approach easily understandable for all stakeholders, in particular customers and end users with the goal to involve them in all steps of the development process.

3 MockupDD Process

In this section we will introduce the MockupDD Process technically. First, we will show how mockups are refined and then translated into presentation models.

Then we will describe how similar refinements are applied in order to obtain navigation models. Finally, we introduce the heuristics applied to derive content models. In all the phases of our process we have chosen to use the UWE methodology because it is representative of an important group of methods, it is based on UML and it has tool support. An overview of the whole process can be observed in Figure 1.

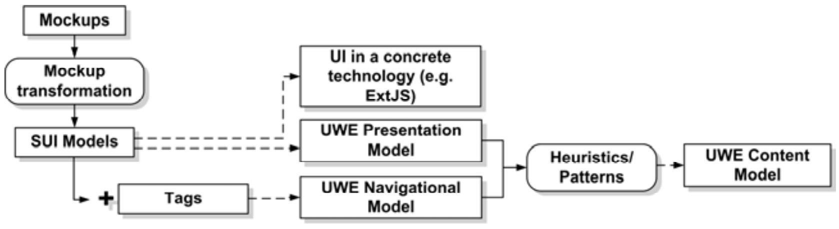
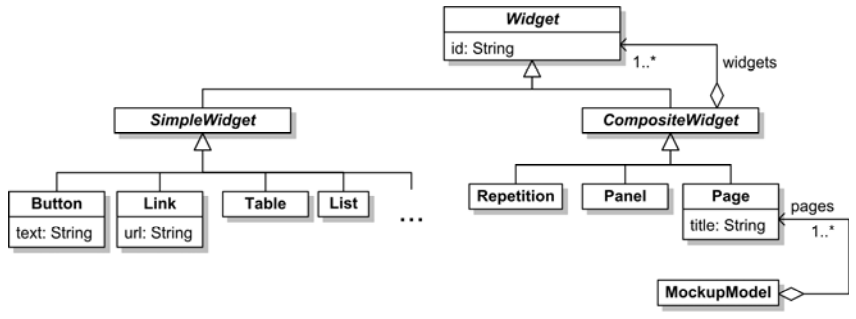


Fig. 1. Mockup-Driven Development (MockupDD) process

3.1 From Mockups to Presentation Models

MockupDD starts the development process by creating UI mockups with a mockup tool. As we have shown in a previous work [18], the resulting mockup files can be parsed and translated to an abstract UI model called *SUI model* (Structural UI Model) that can be in turn translated to presentation models of modern MDWE methodologies through a simple mapping. In Figure 2, SUI metamodel is introduced and the mapping of its elements to UWE Presentation model is shown.



MockupDD Structural UI	UWE Presentation
Panel	PresentationGroup
Panel	Form
Image	Image
Button	Button
Link	Anchor
Label	Text
TextBox	TextInput
Repetition	IteratedPresentationGroup
CheckBox / RadioButton	Selection
Page	Page

Fig. 2. SUI metamodel and UWE mapping

Since mockup tools represent a user interface prototype as a set of unsorted widgets [18], we apply a sequence of different processors that analyzes mockup source structure and outputs the corresponding SUI model. Mockup source processing is done in a pipeline workflow. First, a set of widgets is obtained and validated from a mockup source file using a *Mockup Parser* and a *Validator* respectively. Then, widget composition and repetition is detected with the help of *Hierarchies* and *Repetition Detectors* analyzing the widget set obtained in the previous step. Finally, optimum layout for `CompositeWidget`s is inferred using a *Layout Inferer*. A graphical representation of the whole process can be observed in Figure 3.

3.2 The Tagging Approach

Structural UI models obtained from mockup source through the aforementioned processing represent only the structural view of a web application. In order to add different software features over the existing user interface specification we define the concept of a *tag*. A tag defines a simple but precise specification that is applied over a concrete SUI element and is formed by a name and zero or more textual parameters. Every tag can be applied only over a particular subclass of `Widget` and represents a hint that can result in the derivation of particular MDWE model concepts. Moreover, tags are grouped into *tag sets* that can be combined to construct more complex specification. With the tagging approach we propose a simple, incremental and agile method to model features over previously defined user interface structure (SUI models).

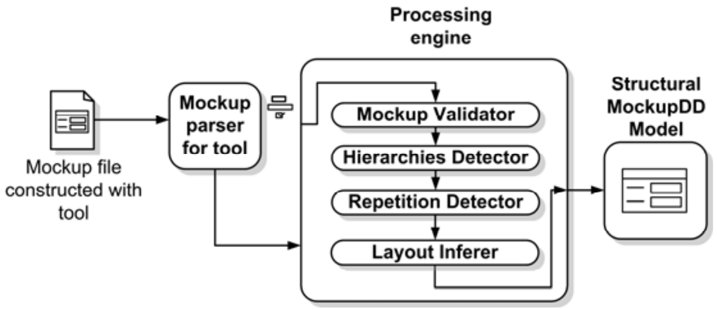


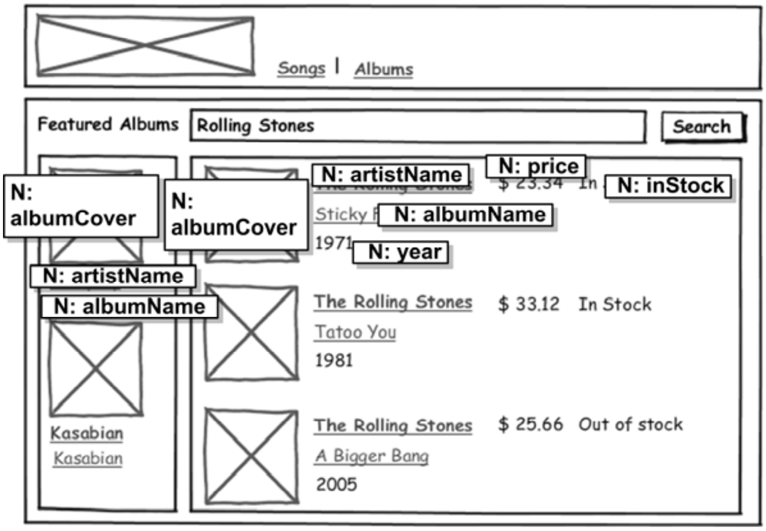
Fig. 3. Mockup processing workflow from original mockup source file until reaching final SUI models

In this paper we introduce *navigation tags* that enrich SUI models in order to derive navigation models. The UI mockup (shown in Figure 4.a) depicts the home page of a music catalogue application (we will call it *Music Portal*) containing a header, a list of featured albums, an album search box and its corresponding search result. Figure 4.b shows the corresponding UWE presentation model that can be obtained applying the previously introduced SUI-to-UWE presentation widget mapping. The *Repetition Detector* processor discovers similar widgets at equal positions and then generates a `Repetition` containing both album lists in the mockup, which are further translated into UWE's `IteratedPresentationGroup`s. By default, generic ids for controls are generated (like `Panel1`, `TextInput1` or `Image1`), but they can be refined using the *Name* tag (denoted with `N:`); these ids are important since they are used to name further MDWE elements. The tagged mockup and resulting UWE presentation model are shown in Figure 4.

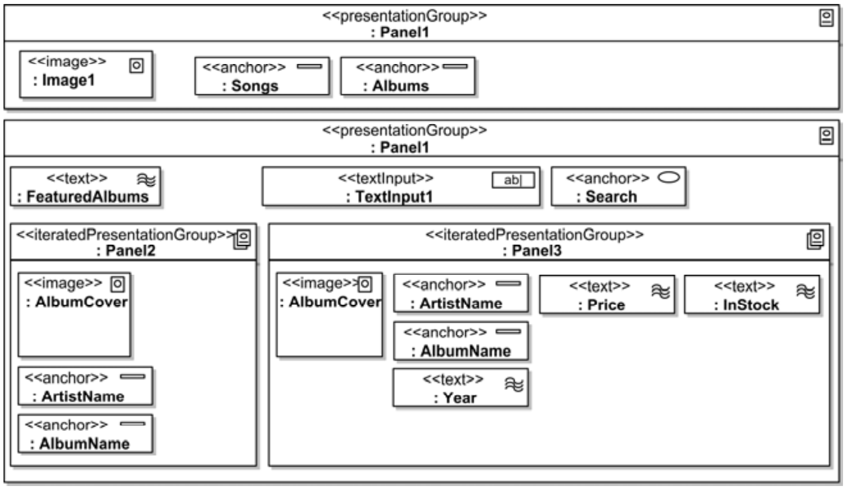
3.3 Deriving Navigational Models

After deriving presentation models, a naive approach to start generating navigation models could be defining one UWE `NavigationNode` (the UWE navigation concept for defining nodes) for each mockup. However, the UWE metamodel defines several navigation elements:

- `NavigationClass`, represents a generic navigable element in the hypertext structure,
- `Menu`, that is used to handle alternative navigation paths,
- `Query`, that is used to retrieve content from a data source, and
- `Index`, that allows selecting one content class instance from a set of instances that have been compiled during previous navigation.



(a) Home page mockup

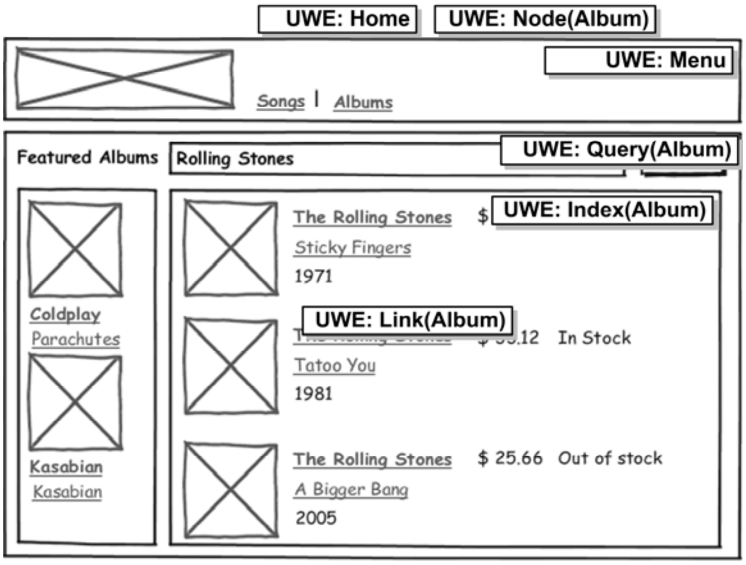


(b) Generated UWE presentation model after applying naming tags

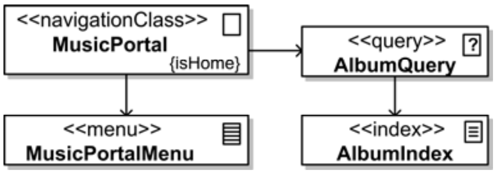
Fig. 4. Deriving an UWE presentation model from a mockup

Additionally, UWE links between navigation elements are expressed through `NavigationLink` instances.

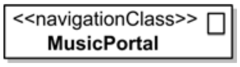
Since we cannot directly infer which UWE navigation element must be used in a mockup as some alternatives are possible (for example, the content of a single mockup may include an UWE `NavigationClass`, a `Menu` and a `Query`), we have defined a second tag set: the UWE navigation tag set. This set contains a tag for every UWE navigation element. Figure 5 shows the resulting tagged mockup and the consequences of tag application in derived UWE navigation model.



(a) Resulting tagged mockup



(c) Navigation model generated with tags

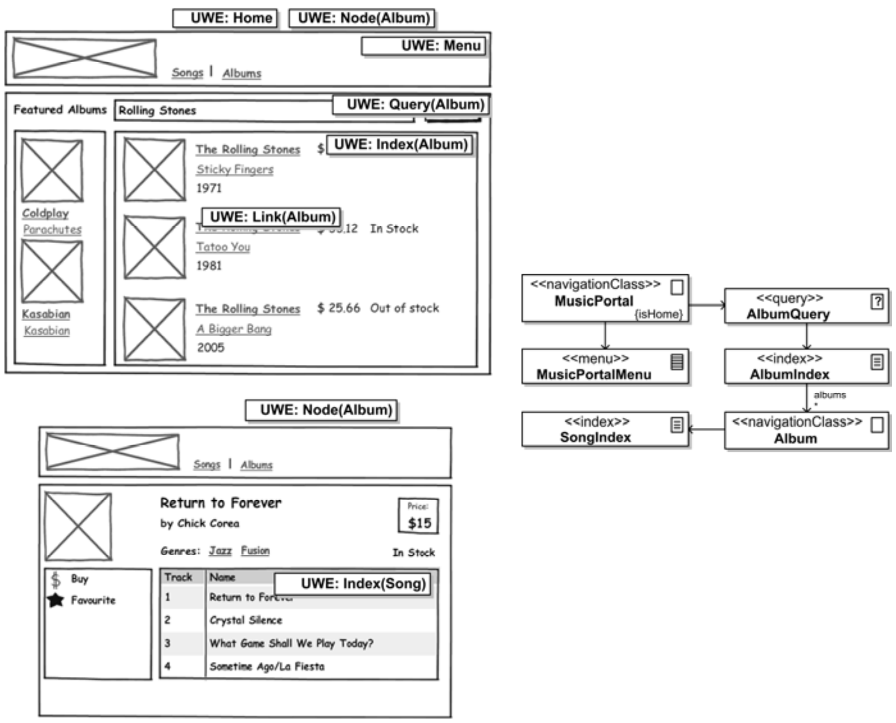


(b) Navigation model generated without tags

Fig. 5. Initial mockup with UWE navigation tags applied and the resulting navigation model.

The UWE navigation tags introduced are the following:

- `Home`: defines that the `NavigationClass` related to the mockup is the home of the navigation model.
- `Node(<nodeId>)`: Assigns an id to the `NavigationClass` related to the mock-up in order to be referenced as the destination of one or more navigation (`Link`) tags.
- `Link(<nodeId>)`: Specifies a navigation link to another `NavigationClass`. A corresponding `Node` tag with the same `<nodeId>` must be specified in order to correctly derive the navigation.
- `Query(<elementId>)` and `Index(<elementId>)` define a `Query` involving elements of type `<elementId>` and the `Index` in which the results of the `Query` are shown.
- `Menu` specifies that the panel over which it is applied is a set of links, a so-called UWE `Menu`.



(a) Home page and album details mockups, property tagged with UWE navigation tags

(b) Resulting navigation generated from mockups in (a)

Fig. 6. Final version of tagged mockups and generated UWE models

When clicking on an album's title in the home page, an UI of the album details will be shown. After being defined, the mockup implementing the added functionality can be joined to the existing model through the aforementioned processing and further tagging, maybe in a new iteration. The *big picture* of the application being modeled can be observed in Figure 6 in which the complete tagged mockups and UWE model generated are depicted. The navigation link between the two existing mockups (SUI models in fact) is expressed through the `Link(Album)` and `Node(Album)` tags in home page and album mockups, respectively.

Some of the transformation rules that we defined (and implicitly applied in the previous example) are schematized in Figure 7.

3.4 Towards a Content Model

Once we have obtained the UWE navigation model, a first version of the content model can be derived by applying some inference rules graphically described in Figure 8. These rules were designed by studying many examples of UWE navigation and content models and discovering recurrent patterns in them.

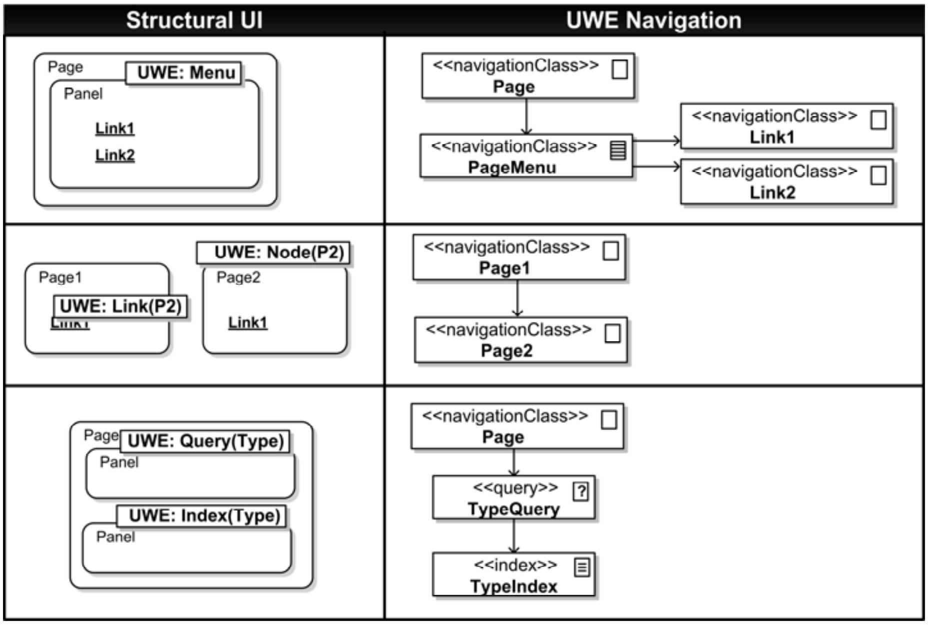


Fig. 7. Transformation rules applied over tagged SUI models to derive UWE features

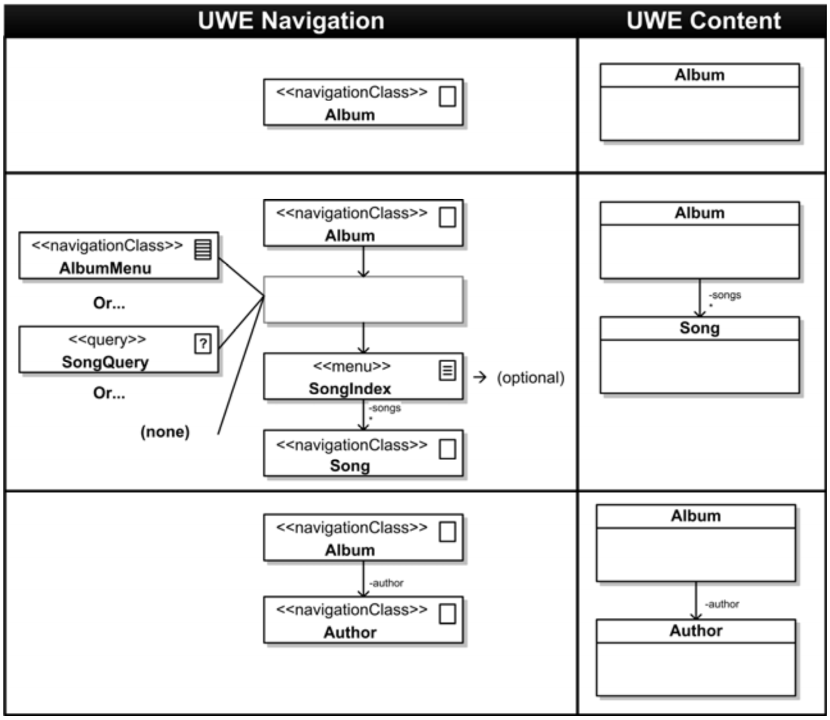


Fig. 8. Some content inference rules to generate UWE Content models from Navigation models

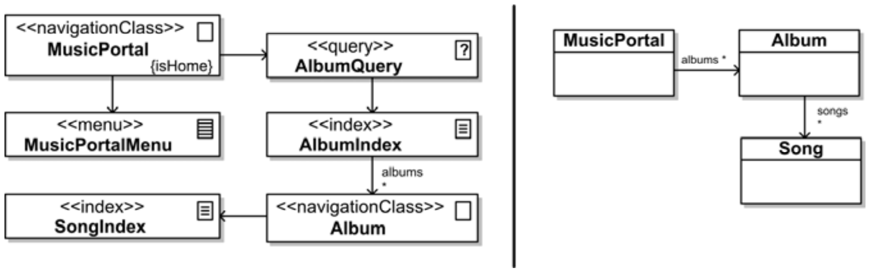


Fig. 9. Inferred UWE content model derived through the application of the introduced rules

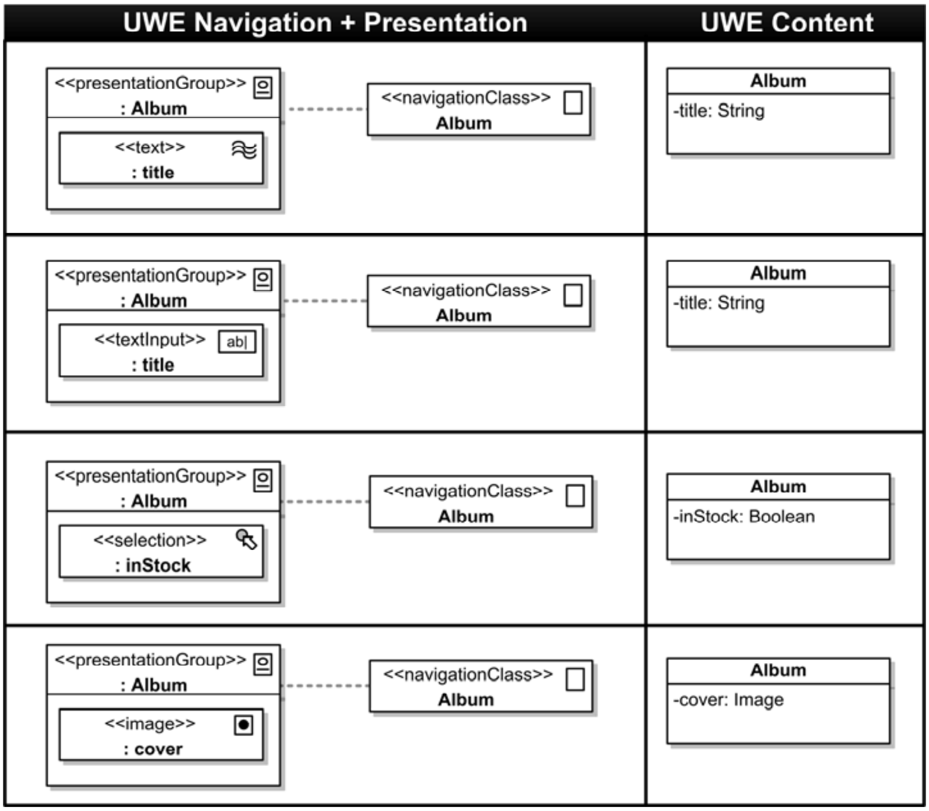


Fig. 10. Attribute inference combining existent navigation and presentation specifications

UWE navigation element names (previously generated using naming and UWE navigation tags) are used to derive the names of the content elements. The resulting UWE content model after the application of the introduced rules over the UWE navigation model of Figure 6.b is shown in Figure 9).

The obtained UWE content models must be refined in order to specify class attributes. As UWE navigation models do not allow more refinement than the features already commented, this information should be taken from other models. Since in

UWE every navigation concept is refined by a presentation specification (e.g., a `PresentationGroup`), and given that we have already derived these models from SUI specifications, we can use this link between models in order to obtain attributes from presentation structure. Rules using this link to infer attributes are graphically described in Figure 10.

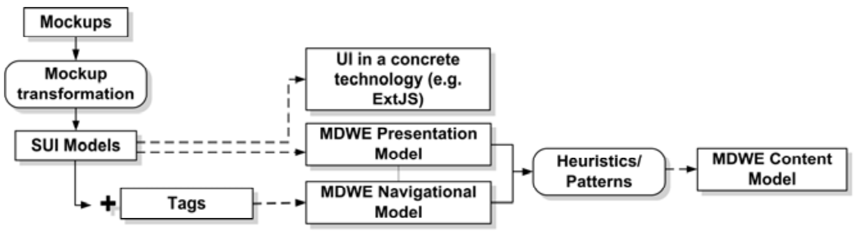
4 Discussion

In this paper we presented an approach that adds agility to existing MDWE methods and we show how it can be applied in the context of UWE. The main intent of our approach is to enable early and constant communication and interaction with end users, a key requirement in agile methodologies. This interaction is facilitated in the early stages of development through the usage of UI mockups as a common language to start the process and discuss requirements; later, during further steps we provide an automatic and fast prototype generation through mockup enrichment and processing with help of our model-driven tooling. Thus, the MockupDD approach changes the traditional MDWE workflow, using presentation models (initially UI mockups) as the starting artifact in the process, and facilitating the incremental and iterative introduction of features through atomic tags over existing models. Since user interface are elements perceived by final customers users, they can be involved early and during every iteration.

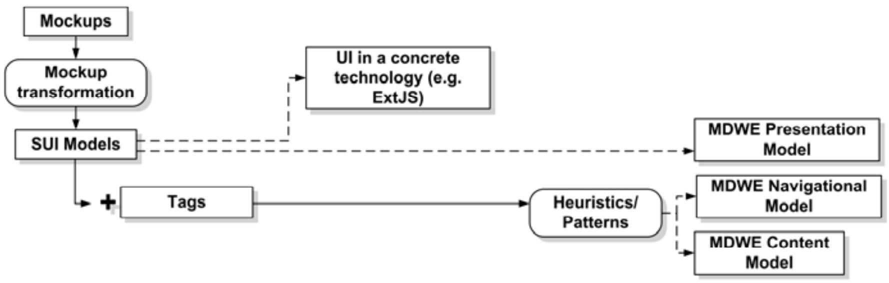
Currently, since MockupDD is intended to provide an agile process to existing MDWE methodologies like UWE, WebML or OOHDM, it only allows specifying features which are present in these approaches, "inheriting" their applicability and limitations in different contexts. According to the current state of our research, many aspects of these methodologies can be generalized in a unified tag set while others must be refined with concrete tag sets, as explained in the following section. Additionally, the automatic derivation process commented in this paper may naturally lead to an imprecise content model, and some thoughtful design changes might be required in order to get to a definitive version. However, even when most design adjustments cannot be fully automated, they can be still predicted. For example, observing the examples in the previous section, an album class in the presentation model might translate into an album class with attributes such as *artistName*, when in fact the content model should have two separate classes for Album and Artist, related to each other. We have observed that many of these inaccurate derivations are usually recurrent, so the required adjustments can be documented (and applied with automatic assistance when possible) just like code refactorings [19].

5 Conclusion and Further Work

We have presented a mockup-based approach (MockupDD) pursuing an inversion of the traditional MDWE process. We decided to start our process with mockups because they are becoming a common tool in agile methodologies to interact and establish a shared view of requirements between customers and developers. Mockups are processed to structured UI models (called SUI) and with the help of the iterative introduction of simple and precise refinements through the so-called *tags* they are easily derived to MDWE presentation and navigation models. Applying a set of inference rules, a first version of MDWE content models can be generated. We have shown the



(a) Workflow of the approach presented in this paper for any MDWE methodology



(b) Workflow of an alternative configuration of the approach in which heuristics and patterns are applied directly over tagged SUI models

Fig. 11. Comparison of the approach presented in this paper (a) and an alternative approach that is being evaluated (b)

approach applied to an example using the UWE methodology. With our approach, we intend to provide an agile methodology based on UI mockups and lightweight specifications to obtain MDWE models, which offer advantages like automatic code generation (increasing software specification productivity) and a high level of abstraction (improving application portability) among others.

Extending the proposed approach to other modern MDWE methodologies like WebML represents a fruitful work path. We are interested in defining a general and methodology-agnostic navigation tag set that will allow us to derive navigation models for a more comprehensive set of MDWE approaches. We are experimenting with the application of heuristics not only at model level as is proposed in this paper, but also directly at the SUIT (SUI plus Tags) level; a comparison of both strategies is depicted in Figure 11. In addition, we are currently working in discovering and cataloging more heuristics and researching about how to define minimum tag sets that provide the highest expressive power and flexibility while preserving the simplicity of the approach.

Since obtained content models likely require to be refactorized, we are interested in developing heuristics to suggest refactoring alternatives to be applied over content specifications. Currently, experiments are being conducted to measure the differences found between MDWE models constructed entirely by hand from mockups and those generated automatically with the proposed tool. The delta found between those models will determine the definition of a catalogue of suggested refactorings and the heuristics implied in the detection of potential *bad smells* in automatically generated models in order to assist the improvement of their quality.

Finally, other approaches that propose *enriching* user interfaces in some way are Portlets and Mashups. While the former represent pluggable user interface elements that can be added to a *portal* page, the second propose to include and combine external services injecting one or more (in this case) UI components into a page. MockupDD propose to discover MDWE elements stereotyping existing user interface elements. However, an interesting branch of our research includes easing *Mashup* building through specific tag sets oriented to include user interface components of external services (e.g., Facebook buttons or comments). Also, we are considering the modularization and further reuse of common elements between mockups (something similar to the *Portlet* approach).

Acknowledgments. This work has been partially sponsored by the EU project ASCENS FP7 257414, and the DFG project MAEWA II, WI 841/7-2.

References

1. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. Computer Networks and ISDN Systems 33(1-6), 137–157 (2000)
2. Gómez, J., Cachero, C.: OO-H Method: Extending UML to Model Web Interfaces. In: van Bommel, P. (ed.) Information Modeling for Internet Applications, pp. 144–173. IGI Publishing, Hershey (2003)
3. Koch, N., Knapp, A., Zhang, G., Baumeister, H.: UML-Based Web Engineering, An Approach Based On Standards. In: Web Engineering, Modelling and Implementing Web Applications, pp. 157–191. Springer (2008)
4. Rossi, G., Schwabe, D.: Modeling and Implementing Web Applications using OOHDM. In: Web Engineering, Modelling and Implementing Web Applications, pp. 109–155. Springer (2008)
5. Wimmer, M., Schauerhuber, A., Schwinger, W., Kargl, H.: On the Integration of Web Modeling Languages: Preliminary Results and Future Challenges. In: Proc. of the 3rd Int. Workshop on Model-Driven Web Engineering (MDWE 2007). CEUR-WS (2007)
6. Harel, D.: Some Thoughts on Behavioral Programming. In: Lilius, J., Penczek, W. (eds.) PETRI NETS 2010. LNCS, vol. 6128, p. 18. Springer, Heidelberg (2010)
7. Seyff, N., Graf, F., Maiden, N.: End-user requirements blogging with iRequire. In: 32nd ACM/IEEE International Conference on Software Engineering - ICSE 2010. ACM Press, New York (2010)
8. Noble, J., Biddle, R., Martin, A.: The XP Customer Role in Practice: Three Studies. In: Agile Development Conference, pp. 42–54. IEEE Computer Society (2004)
9. Ferreira, J., Noble, J., Biddle, R.: Agile Development Iterations and UI Design. In: AGILE 2007 Conference, pp. 50–58. IEEE Computer Society, Washington, DC (2007)
10. Ton, H.: A Strategy for Balancing Business Value and Story Size. In: Agile 2007 Conference, pp. 279–284. IEEE Computer Society, Washington, DC (2007)
11. Kulak, D., Guiney, E.: Use Cases: Requirements in Context. Addison-Wesley (2004)
12. Homrighausen, A., Six, H., Winter, M.: Round-Trip Prototyping Based on Integrated Functional and User Interface Requirements Specifications. Requirements Engineering 7(1), 34–45 (2002)
13. Cohn, M.: User Stories Applied: for Agile Software Development. Addison-Wesley (2004)

14. Moore, J.M.: Communicating Requirements Using End-User GUI Constructions with Argumentation. In: 18th IEEE International Conference on Automated Software Engineering, pp. 360–363. IEEE Computer Society (2003)
15. Panach, J.I., España, S., Pederiva, I., Pastor, O.: Capturing Interaction Requirements in a Model Transformation Technology Based on MDA. Journal of Universal Computer Science 14(9), 1480–1495 (2008)
16. Mukasa, K.S., Kaindl, H.: An Integration of Requirements and User Interface Specifications. In: 6th IEEE International Requirements Engineering Conference, pp. 327–328. IEEE Computer Society (2008)
17. Ricca, F., Scanniello, G., Torchiano, M., Reggio, G., Astesiano, E.: On the effectiveness of screen mockups in requirements engineering. In: 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. ACM Press, New York (2010)
18. Rivero, J.M., Rossi, G., Grigera, J., Burella, J., Luna, E.R., Gordillo, S.: From Mockups to User Interface Models: An Extensible Model Driven Approach. In: Daniel, F., Facca, F.M. (eds.) ICWE 2010. LNCS, vol. 6385, pp. 13–24. Springer, Heidelberg (2010)
19. Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional (1999)

Evaluating Comprehension and Utilization of Variability Aspects in UML-Based Models

Iris Reinhartz-Berger[1], Arnon Sturm[2], and Arava Tsoury[1]

[1] Department of Information Systems,
University of Haifa, Haifa 31905, Israel
iris@is.haifa.ac.il, aravabt@gmail.com
[2] Department of Information Systems Engineering,
Ben-Gurion University of the Negev, Beer Sheva 84105, Israel
sturm@bgu.ac.il

Abstract. In Software Product Line Engineering (SPLE), the ability of a software artifact to be used in different contexts is very essential for productivity. In order to manage and support this ability, different variability modeling methods have been proposed. An important group of such methods are based on UML. These methods typically introduce profiles for specifying mandatory and optional elements, identifying dependencies between elements, and modeling variation points and possible variants. However, the assessment of these methods still lacks. In this work, we have done a first step towards evaluating the comprehension and utilization of variability issues in UML-based models by suggesting a comparison framework which refers to different aspects of variability specification. Based on this framework, we chose a specific UML-based method – ADOM – and examined how advanced information systems students understood and utilized a model specified using this method. The results showed that the different means for specifying variability were understood and utilized only to a limited extent and that variation points were the least comprehensible variability specification means.

Keywords: variability management, software product line engineering, UML, ADOM.

1 Introduction

Software Product Line Engineering (SPLE) [25] deals with two main activities: *domain engineering*, during which a family of software products, termed a product line, is analyzed, designed, and implemented, and *application engineering* in which the particular applications and software products are customized and developed. Accordingly, Bachmann and Clements [3] distinguish between core assets and product artifacts: while ***core assets***, also called domain artifacts, are built to be used by more than one product in the line, ***product artifacts*** are specific parts of the software products. In order to be reusable and suitable to a wide variety of products, the core assets have to specify, besides the commonality, the variability of the given product line. ***Variability*** is defined as the ability of a core asset to be efficiently extended, changed, customized, or configured for use in a particular product artifact [29]. Svahnberg et al. [31] claimed that "development of software product lines relies heavily on the use of variability to manage the differences between products", but "variability is not trivial to manage".

Variability modeling plays an important role in variability management [29]. Reviewing 97 papers that describe variability management methods in SPLE, reported from 1990s to 2007, Chen and Babar [6] conclude that the main corpus of methods focuses on variability modeling and utilizes feature models (33 works) or UML and its extensions (25 works) for this purpose. Feature-oriented methods, such as [20] and [22], support specifying core assets as sets of characteristics relevant to some stakeholders and the relationships and dependencies among them. Variability is specified in terms of mandatory vs. optional features, alternatives, OR features, 'require' and 'exclude' dependencies among features, feature groups, and composition rules. UML-based methods (e.g., [13], [26], [28], [32], and [34]) usually suggest profiles for handling variability-related issues, including specification of mandatory and optional elements, dependencies among elements, variation points, and possible variants [15]. Some UML-based methods suggest extending UML or representing variability aspects orthogonally to "regular" UML models of the product families, e.g., [14]. Other directions, such as applying Domain-Specific Languages (DSL), see for example [17] and [24], are also explored, but the number of works in each such category according to [6] is still very low.

Despite their amenability to be empirically evaluated, relatively minor attention is allocated for the empirical evaluation of SPLE methods in general and variability modeling issues in particular [6]. These studies highlight different aspects in SPLE, including product derivation [30], quality assurance [4, 9], and architecture process activities [1]. However, only one of them [4] refers to the comprehensibility of core assets. This work assesses the maintainability of feature models in terms of analyzability, changeability, and understandability, using a 7-point scale to gather the subjective opinions of the participants on the relevant characteristics. This work does not check the participants' performance in carrying out different domain or application engineering tasks.

As comprehensibility of core assets may affect the correctness of their utilization while creating valid product artifacts, we draw in this paper a general comparison framework for evaluating variability aspects in SPLE methods. This framework, which refers to both specification aids and their utilization during application engineering, is used for better understanding how to specify core assets while conducting domain engineering. According to this framework, the specification aids used in domain engineering are divided into two types: selection and extension. ***Selection variability*** refers to choosing elements from the core asset for a particular product artifact, while performing application engineering. It includes mandatory and optional elements specification, as well as dependencies specification. ***Extension variability***, on the other hand, refers to locations in core assets at which variability may occur (*variation point*) and possible ways to realize this variability in order to create particular product artifacts (*variants*). Extension variability needs to answer different questions, such as: (1) Which variants can be selected at a certain variation point? (2) Can new variants be introduced for a particular product? (3) When will the variability be resolved?

We concentrate here on variability modeling in UML-based methods, due to the popularity and the wide usage of their notation in software engineering in general, the ease of extending them to different product and product line aspects (including behavioral ones, design- and implementation-related aspects, and so on), and the minimal requirements for additional specification (as opposed, for example, to DSLs).

Based on the suggested framework, we reviewed different UML-based methods and selected the Application-based Domain Modeling (ADOM) method whose expressiveness in terms of variability specification includes most of the important concepts. We then carried out an empirical study on information systems students to examine how they understand a domain model (which is a type of core assets) specified using this method. We further checked how the domain model is utilized while creating specific models of applications (i.e., product artifacts in the line). Analyzing the study results, we claim that variability was understood and utilized only to a limited extent and that most of the problems were in understanding variation points and utilizing extension variability.

The remainder of this paper is organized as follows. Section 2 reviews existing methods for modeling variability using UML, whereas Section 3 introduces and exemplifies the ADOM method and its mapping to other UML-based variability modeling methods. Section 4 elaborates on the empirical study, discussing the research questions, the settings, the results, and the threats to validity. Finally, Section 5 concludes and refers to future research directions.

2 Variability Modeling in UML-Based Methods

Most UML-based methods in the field of SPLE define profiles to support the modeling of variability aspects, while some of them suggest modifications to the UML metamodel or specification of a "variability model" orthogonally to the UML models. Table 1 summarizes related work according to the way they specify selection and extension variability and the supported (UML) diagrams. As can be seen, most methods refer to both selection and extension variability. Mandatory (sometimes called kernel) and optional elements are usually specified using dedicated stereotypes, although these stereotypes sometimes refer only to variation points and variants and not to other elements in the core assets. Some works explicitly specify extension 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 «alternative_or», «alternative_XOR», «requires», and «mutux» stereotypes.

In order to evaluate the comprehension and utilization of variability aspects in UML-based methods and to identify factors that may affect these activities, we conducted an empirical study with a method, called Application-based DOmain Modeling (ADOM) [26, 27]. We chose this method over other UML-based methods since it explicitly refers to extension variability and in particular to the selection and the 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 and not just mandatory and optional elements.

Table 1. Comparison of Variability Modeling in UML-based Methods

Method name	**Selection Variability Specification**	**Extension Variability Specification**	**Supported diagrams**
RSEB [18]		Explicit specification of variation points; Variants are specializations of variation points	Use case and class diagrams
PLUS [13]	Distinction between «kernel» (mandatory) and «optional» elements	Explicit specification of variants only («variant»)	All diagrams
Halmans & Pohl [14]	Specification of mandatory and optional variation points via black and white triangles, respectively	Explicit specification of variants («variant»); Explicit relationships between variants and variation points via «include» dependencies	Use case diagrams
Ziadi et al. [34]	Possibility to classify elements as «optional»	Explicit specification of both variation points («variation») and variants («variant»); Variants inherit variation points, which are modeled as abstract classes	Class and sequence diagrams
Alves de Oliveira et al. [1]	Classification of variants as «mandatory», «optional», «alternative_or», «alternative_XOR» for a certain variation point; Dependencies between variants are marked by «requires» and «mutux»	Explicit specification of variation points	Use case, class, and component diagrams
Robak et al. [28]		Explicit specification of variants; Variation points are specified as branches in activity diagrams or components in component diagrams	Component and activity diagrams
SPLIT [8]	The existence attribute indicates whether a variation point is optional or mandatory	Explicit specification of variation points; Relationships between variants and their variation points are classified according to the variability mechanisms («insert», «extend», or «parameterize»)	All diagrams
VPM [32]	Mandatory variation points are specified as 'm', while optional variation points are specified as 'o'	Explicit specification of variants; Variation points are visualized and categorized into four types: Parameterization, Information Hiding, Inheritance, and Callback	All diagrams
ADOM [26, 27]	Associating the stereotype «multiplicity min=m max=n»; Dependencies can be marked as «requires» or «excludes»	Explicit specification of both variation points («variation point») and variants («variant»); Variants inherit variation points; Variation points may constrain the selection and addition of variants	All diagrams

3 The ADOM Method

The Application-based DOmain Modeling (ADOM) method is based on a profile with five stereotypes, presented in Figure 1: «multiplicity», «variation point», «variant», «requires», and «excludes». The *«multiplicity»* stereotype, which is elaborated in [26], is used for specifying the range of elements in a product artifact that can be classified as the same element in the core asset. 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».

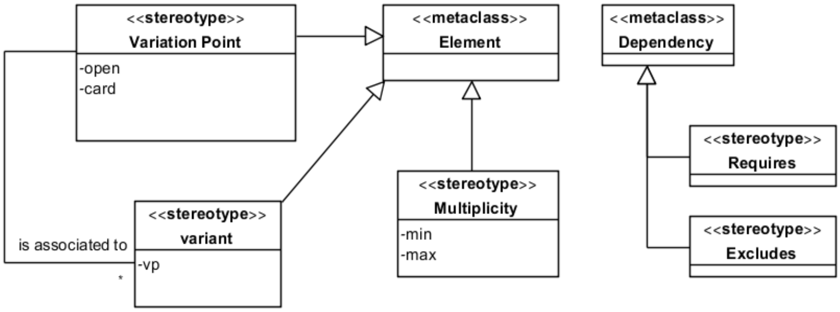


Fig. 1. ADOM's profile

Each element in the core asset may be defined as 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 specific variants that are not specified in the core asset can be added at this point in a particular product 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' (XOR), '1..*' (OR), '0..1' (optional XOR), and '0..*' (optional OR). Note that there are differences between the «multiplicity» stereotype and the cardinality tagged values. A variation point, for example, can be optional 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 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 (which are not specified in the core asset).

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.). In the literature the relationships between a variation point and its variants may take various forms, e.g., inheritance and dependency [7]. Since variation points may specify structural and behavioral aspects that are relevant to all their variants, ADOM supports specifying the relationship between a variant and the relevant variation point via inheritance relationships. When not applicable, i.e., for variation points and variants that are not classifiers, such as attributes, operations, and combined fragments, the relationships between variants and variation points are specified using a tagged value, *vp*, associated to the «variant» stereotype; *vp* specifies the corresponding variation point. Note that the same element in the core asset 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, where A and B are two (optional) elements, 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.

The suggested profile includes in addition a set of rules that specify the allowed combination of stereotypes. For example, the cardinality constraints of a variation point should be feasible in the context of the multiplicity constraints of its variants. However, due to space limitations, the entire set of rules is not presented here.

Figure 2 is a part of a Check-In/Check-Out (CICO) product line specification. The main purpose of this product line is developing applications or software products for checking in and out items. Examples of products in this line are hotel reservation systems, libraries, renting agencies, and version control services. Items that can be checked out have unique identifiers, as well as attributes specifying their statuses, (general) details, check-in details, and check-out details. They are primarily divided into virtual and physical items. Based on this model, handling fees differ according to the item type: physical items must have delay fees handling, whereas damage and lost fee calculations are optional. They further have location details. Virtual items, on the other hand, typically have no location details and no delay, lost, and damage fees. They may need to handle fees when the specified loan policy is violated. According to the associated tagged values of the variation point (open is true and cardinality is optional OR, i.e., '0..*'), a particular application model may include items which are neither physical nor virtual.

In a hotel reservation application, for example, the hotel rooms can be classified as physical items, whereas the different services provided by the hotel can be considered as virtual items. We could specify in the domain model an «excludes» dependency between Physical Item and Virtual Item, indicating that each product in the line may handle either physical or virtual items (but not both), but we did not do that in order to allow creation of CICO software products that handle both physical and virtual items, such as hotel reservation applications.

To check the expressiveness of ADOM with respect to the other reviewed works, we mapped the different stereotypes of ADOM to those of the reviewed methods. Table 2 presents this mapping. As can be seen, the important concepts are explicitly supported by ADOM.

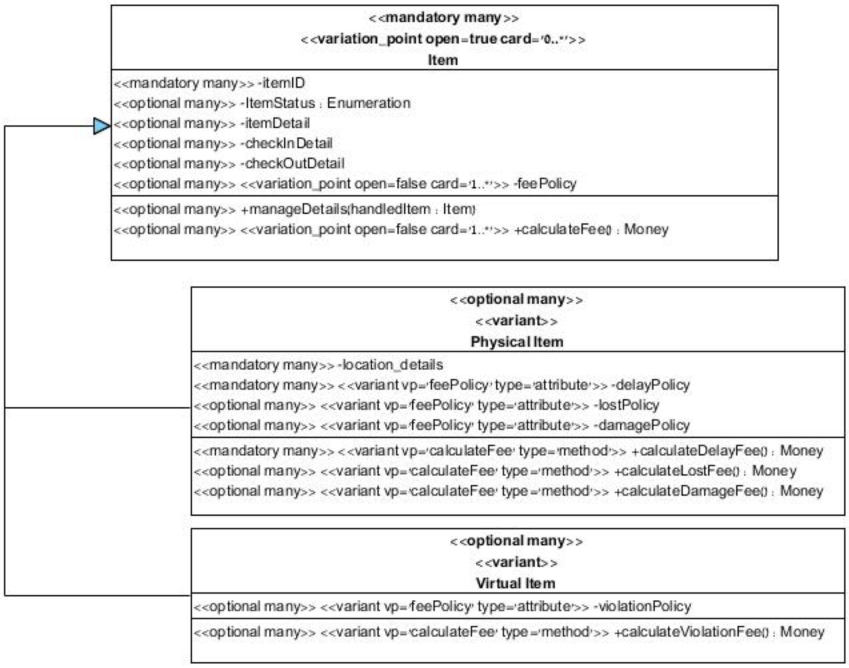

<<mandatory many>>
<<variation_point open=true card='0..*'>>
Item
<<mandatory many>> -itemID
<<optional many>> -ItemStatus : Enumeration
<<optional many>> -itemDetail
<<optional many>> -checkInDetail
<<optional many>> -checkOutDetail
<<optional many>> <<variation_point open=false card='1..*'>> -feePolicy
<<optional many>> +manageDetails(handledItem : Item)
<<optional many>> <<variation_point open=false card='1..*'>> +calculateFee() : Money
<<optional many>>
<<variant>>
Physical Item
<<mandatory many>> -location_details
<<mandatory many>> <<variant vp='feePolicy' type='attribute'>> -delayPolicy
<<optional many>> <<variant vp='feePolicy' type='attribute'>> -lostPolicy
<<optional many>> <<variant vp='feePolicy' type='attribute'>> -damagePolicy
<<mandatory many>> <<variant vp='calculateFee' type='method'>> +calculateDelayFee() : Money
<<optional many>> <<variant vp='calculateFee' type='method'>> +calculateLostFee() : Money
<<optional many>> <<variant vp='calculateFee' type='method'>> +calculateDamageFee() : Money
<<optional many>>
<<variant>>
Virtual Item
<<optional many>> <<variant vp='feePolicy' type='attribute'>> -violationPolicy
<<optional many>> <<variant vp='calculateFee' type='method'>> +calculateViolationFee() : Money

Fig. 2. Specification of Item variation point in ADOM

Table 2. Mapping between ADOM and other UML-based methods

Stereotype in ADOM	Corresponding stereotypes in the related works
«mandatory single», «mandatory many»	«kernel» [13], «mandatory» [1]
«optional single», «optional many»	«optional» [1, 13, 34]
«variation point»	«variation point» [1, 8], «variation» [34], «virtual» [34]
«variant»	«variant» [13, 14, 18, 32, 34]
Cardinality='1..*'	«alternative_or» [1]
Cardinality='1..1'	«alternative_XOR» [1], «insert» [8]
«requires»	«requires» [1]
«excludes»	«mutux» [1]
Inheritance between variation point and variant	«extend» [8], «include» [14]

4 Comprehension and Utilization of Variability Aspects in ADOM

In order to examine the comprehension and utilization of variability aspects in ADOM, we conducted an empirical study, which aimed at checking the following two research questions:

Question #1 (comprehension): Is selection and extension variability as specified in ADOM's models well understood to software modelers and to what extent?

Question #2 (utilization): Is selection and extension variability as specified in ADOM's models well utilized by software modelers and to what extent? Here we examined two common utilizations of core assets, namely validation of a particular product with respect to the relevant core asset and guidance of particular product creation. Both guidance and validation activities are part of application engineering and are conducted while designing particular product artifacts.

The settings, results, and threats to validity of this study are reported below.

4.1 Study Settings

Since it is very difficult to find suitable subjects for such empirical studies – subjects who know the required material and who are willing to devote their time – we conducted the study with 15 advanced (last year) undergraduate and graduate students in an Information Systems program at the University of Haifa, Israel, who took a seminar course on domain engineering during the winter semester of the academic year 2009-2010. All these students had previous knowledge in software systems modeling and specification (this was their third course on this subject), as well as initial experience in industrial projects. Thus, they can be considered comparable to junior software modelers. During the course, the students studied various domain engineering techniques, focusing on the ADOM method and its ability to specify, guide, and validate variability. They further got homework exercises in which they had to model in groups of four students a domain model and four application models in the same domain or product line, using that domain model. Besides this, the students were not explicitly trained towards the experiment's tasks.

The study took place towards the end of the course as a class assignment. This assignment was worth up to 10 points of the students' grades in the course. The students got an ADOM model of a Check-in/Check-out product line. We chose this example since we believe that the students are familiar with different applications in this domain, such as library, car rental, and hotel reservation systems. This way the terminology used for specifying the core asset was not strange or unfamiliar to them. The model included 3 use case diagrams, 3 class diagrams, and 3 sequence diagrams. In each diagram type, one diagram was considered main (or top level), while the others elaborated the different variants of certain variation points. Overall, seven variation points were specified. Two experts checked the models before the experiment took place and especially their correctness and the ability to answer the questions according to the models. The students had to answer questions that referred to comprehension, validation, and guidance of the given model, as described in the next sub-sections[1].

4.2 Comprehension Questions and Results

The first task, which referred to research question #1, included 14 true/false questions regarding the given CICO model. Examples of questions in this part are:

[1] The complete questionnaire can be found at http://mis.hevra.haifa.ac.il/~iris/research/SPLEeval/

A product in the line may have the possibility to reserve items. In case the product supports this functionality, both loaners and workers should be able to perform it. (a use case diagram related question)

A loaner may handle his/her lending by performing and modifying it. The product should enable recording these operations, and in particular the times in which they occur. (a class diagram related question)

A product in the line may get the item details in a check out scenario by finding this item through the collections it belongs to. (a sequence diagram related question)

For this task, the answers were checked according to a pre-defined solution. For each question we analyzed the answer correctness by examining both the final answer (true/false) and the explanation. An incorrect answer or explanation scored 0, a fully correct answer or explanation scored 1 point, and a partially correct explanation scored 0.5. We grouped the various questions according to the variability types they referred to. Table 3 summarizes the average scores achieved by the students in each category, divided according to the diagram types. For enabling comparison, all scores were converted to percentages that represent the success or correctness rate. As can be seen, the explanation scores were lower than the answer scores in the various question categories. We believe that the reason for that is the difficulty of the students to explicitly point out the reasons for their answers and to justify them.

When analyzing the selection variability, we noticed that the students performance in the use case diagrams were very low. Since this type of diagram is relatively simple, we deeply analyzed the questions in this category and figured out that most of them referred to a model segment rather than to particular elements. The students had problems to find the right answers since they had to gather information from different sources (the elements that composed the segment).

Table 3. Results of the first part (comprehension)

Method		Answer	Explanation	Comments
Selection Variability	Use Case	40.0%	40.0%	The questions in this category referred to segments in the UC diagrams and not to individual elements.
	Class	76.7%	53.3%	
	Sequence	80.0%	63.3%	
	Overall	62.7%	50.0%	
Extension Variability	Use Case	51.1%	44.4%	All questions in this category referred to variation points and how to select particular variants.
	Class	55.6%	32.2%	All questions in this category referred to particular variants rather than to variation points.
	Sequence	46.7%	28.9%	The questions in this category referred to both variants and variation points.
	Overall	51.1%	35.2%	Overall, the performance of variants-related questions was better than that of variation points-related questions.

When analyzing the extension variability results, we found out that the performance in all diagram types was relatively low, 46%-56%. Trying to reach a deeper insight, we separately examined the comprehension of variation points and the comprehension of variants and we found out that the performance in variant-related questions was better than the performance in variation point-related questions (56% vs. 45% on the average). Our conjecture regarding this observation is that variation points are more abstract, usually refer to several elements (variants) and include information regarding the way to realize the variability. Thus, their specification may be more difficult to understand than that of variants, which are more concrete and focus on particular elements.

4.3 Validation Questions and Results

The second task, which partially referred to research question #2, included a model of a hotel reservation application, which is in the scope of the Check-In/Check-Out product line. Following the domain and application models the students were requested to identify places where the hotel reservation model violates the product line constraints. Due to difficulties in defining and understanding the semantics of sequence diagrams [33], we chose to concentrate in this part on functional and structural aspects only. Thus, the hotel reservation model consisted of a use case diagram and a class diagram. The violations referred to both selection and extension variability. Differently from the first task, this part was open and the students were not notified regarding the numbers of violations or their distribution.

For checking this task, we prepared a list of 9 mistakes (or inaccuracies) in the hotel reservation application, according to the corresponding domain model. As we had to analyze the relevance of the mistakes found by the students with respect to our predefined list, we measured the performance in this part in terms of precision, recall, and F-measure [23], which are standard metrics for measuring the relevance of retrieved items (mistakes or inaccuracies in our case). Precision measures the fraction of items in the answer that are correct, while recall measures the fraction of expected items that are in the answer. Since the two above metrics measure different concepts, we used F-measure, which is a derived measure defined as the harmonic mean of precision and recall [23]:

$$\mathrm{F-measure} = \frac{2 * \mathrm{precision} * \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

Table 4 summarizes these measurements according to the two variability types, selection and extension, and the two diagram types. As can be clearly seen through the F-measurement, the results are poor, especially when looking at the low values of recall (less than 20% in the overall). However, errors that referred to selection variability were much easier to find than errors that referred to extension variability. Although the recall in the selection variability category was quite similar in use case and class diagrams (~27% and ~36%, respectively), the precision was quite different: the precision of finding errors in use case diagrams was lower than the precision of finding errors in class diagrams. One possible explanation to this observation is that the students were more familiar and experienced with class diagrams (a case that is also reported in [11]). Another explanation is that the use case diagram referred to two basic operations: check-out, i.e., borrowing an item, which is called check-in in the hotel application, and check-in, i.e., returning an item, which is termed check-out in the hotel application. The swapping of these terms in the particular application caused a lot of difficulties to the students.

Table 4. Results of the second task (validation)

Method		Precision	Recall	F-measure
Selection Variability	**UC**	29.86	26.67	28.17
	Class	58.93	35.56	44.35
	Overall	45.92	32.00	37.72
Extension Variability	**UC**	100.00	3.33	6.45
	Class	100.00	3.33	6.45
	Overall	100.00	3.33	6.45
Overall		47.58	19.26	27.42

In the extension variability category, only one student found one error in the use case diagram and another student found a single error in the class diagram. In general, we believe that the reason for this result is the fact that the mistakes and inaccuracies were not obvious and spanned within different model elements.

Analyzing the students' explanations in this part, we found three sources of problems. First, the students had difficulties in mapping the elements of the hotel reservation application to the CICO elements. As this mapping may reveal anchors for validating particular product artifacts with respect to core assets, these difficulties also prevent the students from correctly identifying problems that relate to selection variability. Second, checking each element in the application against its corresponding core asset's element, which may appear in several diagrams (e.g., in the case of a variation point), is a difficult and exhaustive task that may yield inconsistencies in the provided answers. This problem, which probably affected the low recall in all categories and especially in the extension variability category, supports the need for well established processes, techniques, and tools for validating particular product artifacts against their relevant core assets. Finally, the openness of this task caused the students to find problems that were not actually mistakes, while trying to generate complete lists of problems, partially explaining the low precision in the selection variability category.

4.4 Guidance Questions and Results

Finally, in the third part, which also referred to research question #2, the students got a list of requirements for a library application, which also belongs to the Check In/ Check Out product line. Based on the domain model and the library requirements, the students were asked to develop a specific model for the library application. Examples of requirements in this part are:

Some of the copies can be borrowed online, without visiting the library and getting the actual copies. However, a loaner can get these copies also physically.

Online borrowers may not be members of the library. In this case, they have to pay for each online borrowing, before the copy is borrowed. Members can pay for their online borrowing when the due date of the borrowing arrives or when the online copy is expired.

If the book or journal copy has been damaged during the borrowing, the librarian will ask the system to calculate the damage charges for the specific book or journal. For each book or journal, there are 3 damage fees: minor damage fees, medium damage fees, and major damage fees.

In this part, we examined how the aids for specifying selection and extension variability were utilized for guiding the creation of models for the library application. As different models could be provided for the same set of requirements, we graded the models correctness and completeness with respect to the library requirements.

Analyzing the results, we found out that the specification of selection variability guides the students to a large extent. Thus, in the following we chose to concentrate on the extension variability specification. In particular, we separately analyzed how variation points and variants are utilized for guiding application development. Table 5 summarizes the results achieved in the third part of the study with respect to the extension variability. The different numbers indicate the numbers of students (out of 15) correctly/wrongly specified each variation point and its related variants. Variation point specification refers to the selection and addition of the different variants (as expressed in the open and cardinality tagged values), whereas variant specification refers to the structure of the relevant variants as specified in the domain model or inherited from the variation point specification. All the variation points were required for completely specifying the library application.

We further examined the correlation between the success of utilizing a variation point and the success of utilizing its variant(s) using the Phi coefficient which is suitable for analyzing the correlation between two binary variables [12]. As can be seen, in most cases there is a correlation between the success of utilizing variation points and their associated variants. Two exceptions to this conclusion are the loaner and the item variation points. In the loaner case, which appeared in the use case diagrams, the variation point specified a cardinality of exactly 3 (i.e., a loaner can be either physical or virtual, either member or occasional loaner, and either cooperation or private). Almost all students (14 out of 15) did not enforce this constraint. However, all of them succeeded in modeling the different loaners (variants) and their relations to use cases. Regarding items, there was confusion between the actual items (copies) and their types (e.g., books). Nevertheless, when modeling a particular variant, the students followed all the provided guidance.

Table 5. Results of the third task (guidance)

#	**Variation point (VP) name**	**Diagram type**	**Element type**	**VP Utilization**	**Variant Utilization**	**Correlation (Phi)**
1	Loaner	Use case	Actor	1/15	15/15	-1.00 (est.sig=.157)
2	Check-In Item	Use case	Use case	13/15	13/15	1[*]
3	Item	Class	Class	4/15	15/15	-1.00 (est.sig =.157)
4	Fee	Class	Attribute	1/15	1/15	1[*]
5	calculateFee	Class	Operation	4/15	4/15	1[*]
6	Handle	Class	Association	15/15	15/15	1[*]
7	getItemDetails[**]	Sequence	Comb. Fragment	9/13	9/13	1[*]

[*] The Phi coefficient cannot be computed due to exact match.

[**] Two students completely ignored this variation point and thus were omitted from the statistical analysis of the utilization correlation.

Another observation that can be made on the results presented in Table 5 is that the guidance provided by variation points was less considered than the guidance provided by the variants. With respect to the variation point guidance, only 2 variation points were used by most of the students (#2 and #6 in the table), 1 variation point was used by many students (#7) and the other variation points were used to a very limited extent (#1, #3, #4, and #5). With respect to the variant guidance, most (5) variants were used by most of the students (#1, #2, #3, #6, and #7), while the other 2 variants were used to a very limited extent (#4 and #5). It can also be seen that low level elements, such as attributes and operations, were less handled with regard to both variation point and variant specifications (#4, #5, and #7). A possible reason for this may be that the students concentrated on main, top level concepts and the relationships between them, neglecting low level details.

4.5 Threats to Validity

In our study, we found out that variability is understood and utilized to a limited extent and that extension variability is more difficult to understand and utilized. Nevertheless, these results should be discussed under the following validity threats that are explained below, along with the actions we made to overcome these threats.

The first threat concerns the number and the type of employed subjects. We carried out the study with only 15 students, due to the difficulty to find suitable subjects who are willing to devote their time. However, these students got very good grades in software systems modeling and domain engineering throughout the entire course (an average of 91). Furthermore, they were trained with the suggested profile and motivated (by 10 points of their final grades) to perform the requested tasks well. Kitchenham et al. [21] 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. In our case, the expected users of the method are software developers, or more accurately modelers, who are not necessarily familiar with the given domain. Thus, the selected students may represent the expected users. Only further studies may confirm or disconfirm whether our results can be generalized to more experienced subjects (e.g., software developers and modelers in industry).

The second threat we need to refer to is the simple tasks and models used in the study (this was done in order to adjust the tasks to the capabilities of the subjects within the time frame). On this issue, we can say that the models were carefully built referring to different variability-related challenges. However, we aim at keeping the models simple, yet realistic. Note that despite the small models, essential problems were encountered, making it interesting to check how things would result with real systems and large domains.

Finally, the study was carried out with a specific method, ADOM. Although this method was selected after a careful examination of different UML-based methods (see the resultant mapping in Table 2), comparative analysis needs to be done in order to check the comprehension and utilization capabilities of different methods in this category. Such analysis can be done by defining a set of criteria, similar to the ones listed, for example, in [10] and [16], and examining how the different methods satisfy these criteria. Complementarily, this analysis can use comparative empirical evaluation techniques, involving several UML-based methods.

5 Summary and Future Work

Variability specification is important as it may help create valid applications in certain domains. In this paper, we provide some empirical evidence to the difficulties in comprehending and utilizing variability specified in domain models and potential sources for these difficulties. We found out that variability was understood and utilized to a limited extent. The enforcement of variability-related constraints, especially those that refer to extension variability, and the creation of specific models in the domain were very difficult tasks. We found that it was especially difficult to utilize a domain model for validating a specific application model in the domain. Furthermore, the guidance of the domain model with respect to variation points was very limited when creating a new application model. In general, we found out that variation points were less understood and utilized than selection variability and variants. These findings call for developing explicit and focused methods and tools that will help comprehend variability aspects, as well as the allowed and forbidden options of applications, in certain domains.

In the future, we will replicate the empirical study on larger classes of trained domain engineering students and software modelers and develop case studies of larger domains. Moreover, we intend to revise the tasks and use them for comparing additional variability modeling methods. These can be done in the era of UML-based methods, but other alternatives also exist. For example, a possible solution may be the adoption of a variability language such as the emerging standard of CVL [17] or the utilization of feature-oriented approaches as model composition [19].

We also plan to analyze the results with respect to cognitive theories, such as those presented in [5], in order to gain further understanding on the ways according to which variability modeling methods should be designed.

References

1. Ahmed, F., Capretz, L.F.: The software product line architecture: An empirical investigation of key process activities. Information and Software Technology 50, 1098–1113 (2008)
2. Alves de Oliveira, E., Gimenes, I.M.S., Huzita, E.H.M., Maldonado, J.C.: A Variability Management Process for Software Products Lines. In: Centre, I.B.M. (ed.) for Advanced Studies Conference Archive, Proceedings of the 2005 Conference of the Centre for Advanced Studies on Collaborative Research Table of Contents, Toranto, Ontario, Canada, pp. 225–241 (2005)
3. Bachmann, F., Clements, P.C.: Variability in Software Product Lines. Technical Report CMU/SEI-2005-TR-012 (2005), `http://www.sei.cmu.edu/library/abstracts/reports/05tr012.cfm`
4. Bagheri, E., Dasevic, G.: Assessing the Maintainability of Software Product Line Feature Models using Structural Metrics. Software Quality Journal (2011), doi:10.1007/s11219-010-9127-2
5. Bajaj, A., Rockwell, S.: COGEVAL: Applying Cognitive Theories to Evaluate Conceptual Models. Advanced Topics in Database Research (4), 255–282 (2005)

6. Chen, L., Babar, M.A.: A systematic review of evaluation of variability management approaches in software product lines. Information and Software Technology 53, 344–362 (2011)
7. Clotet, R., Dhungana, D., Franch, X., Grunbacher, P., Lopez, L., Marco, J., Seyff, N.: Dealing with Changes in Service-Oriented Computing Through Integrated Goal and Variability Modelling. In: Proceeding of the Second International Workshop on Variability Modelling of Software-intensive Systems (VaMoS 2008), pp. 43–52 (2008)
8. Coriat, M., Jourdan, J., Fabien, B.: The SPLIT method: building product lines for software-intensive systems. In: Proceedings of the First Conference on Software Product Lines: Experience and Research Directions: Experience and Research Directions, pp. 147–166 (2000)
9. Denger, C., Kolb, R.: Testing and Inspecting Reusable Product Line Components: First Empirical Results. In: Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering, pp. 184–193. ACM (2006)
10. Djebbi, O., Salinesi, C.: Criteria for Comparing Requirements Variability Modeling Notations for Product Lines. In: The Fourth International Workshop on Comparative Evaluation in Requirements Engineering (CERE 2006), in Conjunction with RE 2006 (2006)
11. Dobing, B., Parsons, J.: How UML is used. Communication of the ACM 49(5), 109–113 (2006)
12. Edwards, A.L.: The Phi Coefficient. In: An Introduction to Linear Regression and Correlation, pp. 68–72. W. H. Freeman, San Francisco (1976)
13. Gomaa, H.: Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. Addison-Wesley Professional (2004)
14. Halmans, G., Pohl, K.: Communicating the Variability of a Software-Product Family to Customers. Software and Systems Modeling 2(1), 15–36 (2003)
15. 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)
16. Haugen, Ø., Møller-Pedersen, B., Oldevik, J.: Comparison of System Family Modeling Approaches. In: Obbink, H., Pohl, K. (eds.) SPLC 2005. LNCS, vol. 3714, pp. 102–112. Springer, Heidelberg (2005)
17. Haugen, Ø., Møller-Pedersen, B., Oldevik, J., Olsen, G.K., Svendsen, A.: Standardized Variability to Domain Specific Languages. In: Proceedings of the 12th International Conference on Software Product Line, pp. 139–148 (2008)
18. Jacobson, I., Griss, M., Jonsson, P.: Software Reuse-Architecture, Process and Organization for Business Success. ACM Press, New York (1997)
19. Jayaraman, P., Whittle, J., Elkhodary, A.M., Gomaa, H.: Model Composition in Product Lines and Feature Interaction Detection Using Critical Pair Analysis. In: Engels, G., Opdyke, B., Schmidt, D.C., Weil, F. (eds.) MODELS 2007. LNCS, vol. 4735, pp. 151–165. Springer, Heidelberg (2007)
20. 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)
21. 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)

22. 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)
23. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
24. Mernik, M., Heering, J., Sloane, A.M.: When and How to Develop Domain-Specific Languages. ACM Computing Surveys (CSUR) 37(4), 316–344 (2005)
25. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer, Heidelberg (2005)
26. Reinhartz-Berger, I., Sturm, A.: Utilizing Domain Models for Application Design and Validation. Information and Software Technology 51(8), 1275–1289 (2009)
27. Reinhartz-Berger, I., Tsoury, A.: Experimenting with the Comprehension of Feature-Oriented and UML-Based Core Assets. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPMDS 2011 and EMMSAD 2011. LNBIP, vol. 81, pp. 468–482. Springer, Heidelberg (2011)
28. 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)
29. Sinnema, M., Deelstraa, S.: Classifying Variability Modeling Techniques. Information and Software Technology 49(7), 717–739 (2007)
30. Sinnema, M., Deelstra, S.: Industrial validation of COVAMOF. Journal of Systems and Software 81(4), 584–600 (2008)
31. Svahnberg, M., Van Gurp, J., Bosch, J.: A Taxonomy of Variability Realization Techniques. Software – Practice & Experience 35(8), 705–754 (2005)
32. Webber, D., Gomaa, H.: Modeling variability in software product lines with variation point model. Science of Computer Programming 53, 305–331 (2004)
33. Xiaoshan, L., Zhiming, L., Jifeng, H.: A formal semantics of UML sequence diagram. In: Australian Software Engineering Conference, pp. 168–177 (2004)
34. Ziadi, T., Hélouët, 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)

OLAP on Complex Data: Visualization Operator Based on Correspondence Analysis

Sabine Loudcher and Omar Boussaid

ERIC laboratory, University of Lyon (University Lyon 2),
5 avenue Pierre Mendes-France, 69676 Bron Cedex, France
(sabine.loudcher,omar.boussaid)@univ-lyon2.fr

Abstract. Data warehouses and Online Analysis Processing (OLAP) have acknowledged and efficient solutions for helping in the decision-making process. Through OLAP operators, online analysis enables the decision-maker to navigate and view data represented in a multi-dimensional manner. But when the data or objects to be analyzed are complex, it is necessary to redefine and enhance the abilities of the OLAP. In this paper, we suggest combining OLAP and data mining in order to create a new visualization operator for complex data or objects. This operator uses the correspondence analysis method and we call it VOCoDa (Visualization Operator for Complex Data).

Keywords: OLAP, complex data, visualization, factor analysis.

1 Introduction

Data warehouses and Online Analysis Processing (OLAP) have recognized and effective solutions for helping in the decision-making process. Online analysis, thanks to operators, makes it possible to display data in a multi-dimensional manner. This technology is well-suited when data are simple and when the facts are analyzed with numeric measures and qualitative descriptors in dimensions. However, the advent of complex data has questioned this process of data warehousing and online analysis.

Data are said to be complex when they are:

– represented in various formats (databases, texts, images, sounds, videos...);
– diversely structured (relational databases, XML documents...);
– originating from several different sources;
– described through several channels or points of view (a video and a text that describe the same meteorological phenomenon, data expressed in different scales or languages...);
– changing in terms of definition or value (temporal databases, periodical surveys...).

Complex data often contain a document, an image, a video, ..., and each of these elements can be described and observed by a set of low-level descriptors or by semantic descriptors. This set of elements can be seen not only as complex data but also as a complex object. A complex object is a heterogeneous set of data, which, when combined, form a semantic unit. For instance, a patient's medical record may be composed by heterogeneous elements (medical test results, X-rays, ultrasounds, medical past history, letter from the current doctor, ...) and is a semantic unit. It is a complex object.

As said above, warehousing and online analytical processes must be modified in the case of complex objects. In this paper, we focus on the visualization of complex objects. The problem of storing and modeling complex objects is discussed in other articles [6,5]. The purpose of online analysis is to (1) aggregate many data to summarize the information they contain; (2) display the information according to different dimensions (3) navigate through data to explore them. OLAP operators are well-defined for classic data. But they are inadequate when data are complex. The use of other techniques, for example data mining, may be promising. Combining data mining methods with OLAP tools is an interesting solution for enhancing the ability of OLAP to analyze complex objects. We have already suggested extending OLAP capabilities with complex object exploration and clustering [2,3].

In this paper, we are concerned with the problem of the visualization of complex objects in an OLAP cube. By this means, we aim to define a new approach to extending OLAP capabilities to complex objects. With the same idea of combining data mining and online analysis, some works suggest using *Visual Data Mining* technology for visually and interactively exploring OLAP cubes. Maniatis *et al.* list possible representations for displaying a cube and offer the CPM model (*Cube Presentation Model*) as a model in an OLAP interface [11]. The CPM model borrows visualization tools from the field of the HMI (Human Machine Interface). Unfortunately, these works do not take complex objects into account. In a cube of complex objects, the facts are indeed complex objects, and the dimensions can include images, texts, descriptors, ... and OLAP measures are not necessarily numeric. Given these characteristics, standard visualization tools are not necessarily well-suited and should be adapted. To do this, we use Correspondance Analysis, a factor analysis method known in data mining, because it makes it possible to visualize complex objects while highlighting interesting facts for analysis [4,13]. Correspondance Analysis represents objects by projecting them on to factor axes. In a previous paper, we laid the foundations for this proposal [12]. In this paper, we complete and improve our first proposal by taking into account the measure to visualize complex objects, using indicators to make interpretation easier. We thus offer a comprehensive approach and a new OLAP operator entitled VOCoDa (*Visualization Operator for Complex Data*).

To illustrate our point of view, we complete the previously used case of researchers' publications, presented in the next section. A publication can be seen as a complex object, or as a semantic entity. We plan to analyze publications according to their authors, national or international range, support such as a conference or a journal, etc. We aim to observe the diversity of the themes in which researchers publish and the proximity of authors when they are working on the same themes. We need a number of OLAP operators which explore the semantic content of publications. We do not want to limit ourselves to classic online analysis, which provides arithmetic operators for aggregating numeric data.

This paper is organized as follows. First, we present the running example of the analysis of scientific publications in a research laboratory. Then in section 3, we continue by positioning and presenting our approach. In section 4, we develop our approach through analysis of the publications. We end this paper with a conclusion focusing on certain future perspectives (section 5).

2 Running Example

A scientific paper can be viewed as a complex object, and thus as a semantic unit. It consists of items such as the year, the support (the name of the journal or the conference) with a national or international range, and the document itself. The document contains the name of the authors, the title, a summary and the body/content of the paper. An author can be represented by his name, by his picture (image), as well as by other attributes such as his status (full professor, associate professor, PhD student,...). Here, we observe publications as complex objects. To handle these semantic entities, we therefore need an adapted modeling and analysis tool.

In addition to standard descriptors such as year, type, authors, number of pages, etc., the user may also want to analyze the semantic content of the publication, i.e. the topics and content of the publication. The semantic content of the publication must be taken into account when modeling and carrying out an analysis. Let us suppose that the user wants to analyze publications according to the first author, support, year, content and topics of the paper.

As we are in the context of data warehouses, the first step is to model publications in a multi-dimensional manner. In a multidimensional model, we represent facts that are analyzed through measures, usually numerical, and dimensions representing analysis axes [10,7]. These dimensions include descriptors of the facts, and can be grouped into hierarchies with different granularity levels. In our multidimensional modeling, the fact is the publication (cf. figure 1). We choose to observe it in relation to several dimensions: *time*, *authors*, *support*, *keywords* of the publication and *document*. In the *time* dimension, the *year* represents the finest level. It can be grouped by *period*. The *authors* dimension contains two hierarchical levels that aggregate *authors* according to their *status*. The dimension *support* provides the name of the conference or journal or book, as well as the volume, and number of pages. The *type* level aggregates support according to type (conference, journal, chapter, book, ...). Types of publication may be aggregated according to their national or international *scope*. The last two dimensions are semantic dimensions with the keywords associated to the publication and the document itself. *Keywords* can be grouped by *themes* then by *metathemes* (families of themes). Publications are written in English or French, but all keywords are in English. The *document* dimension contains the title, the abstract and the body of the paper.

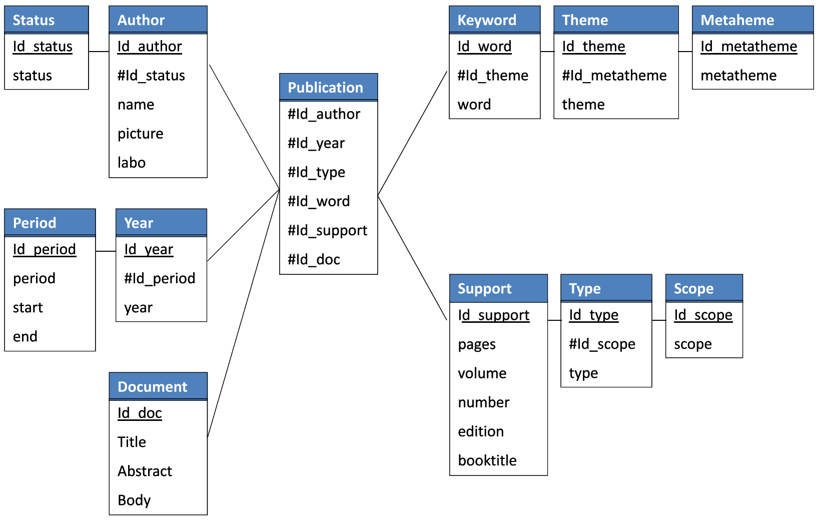


Fig. 1. Multidimensional modeling of publications

Another point of view is that the facts could also be Authors, analyzed according to their publications, status, etc.; in the same way, keywords could be the facts, analyzed according to authors, years, and so on. In multidimensional modeling, measures and dimensions are interchangeable [8].

In this model, there are therefore "classic" dimensions with hierarchies, and semantic dimensions consisting of a hierarchy of concepts (keywords, themes and metathemes) and the document itself.

In multidimensional modeling of publications, difficulties are encountered when it comes to defining one or more numeric measures. In our model, the fact being observed was the publication and it was a combination of all dimensions without a measure. Generally, in case like this where there are no measures , the aggregation function COUNT can be used to count the facts. This solution is always possible in our case, but it is not sufficient because the analysis which follows is too poor.

In cases with no measure, we use the function COUNT to count publications, but we seek other means to analyze publications in order to discover thematic proximity, authors who work together,... We consider a publication as a complex object and we are looking for a way to make a semantic analysis. We propose a visualization of complex objects which takes the semantic content of objects into account. This explains our decision to use a factor analysis method for the visualization of complex objects. This new visualization method fits completely with the online analysis of complex objects.

3 Positioning and Principle

Generally, OLAP interfaces represent a cube as a table, or cross-table (contingency table). In an attempt to exceed the limits of standard interfaces, more advanced tools offer visual alternatives to represent the information contained in a cube, and to interactively browse the cube. Among existing alternatives, there are hierarchical visualizations (trees of decomposition, ...), multi-scale views, interactive scatter plots, and so on. Vinnik *et al.* suggest an improved standard OLAP interface that gives users a dynamic decomposition of the cube with hierarchical visualization techniques [17].

For a better visualization of information, Sureau *et al.* suggest rearranging the modalities of a level according to heuristics, based on distance between the elements in a dimension or according to a genetic algorithm [16]. Genetic algorithms make it possible to modify the cube, thus maximizing evaluation viewing criteria. However, heuristic algorithms, are more competitive than genetic algorithms because they give a better ratio between improved visualization and computation time. In the future, authors hope to integrate all their reorganization algorithms into their virtual reality platforms VRMiner for a complete OLAP environment.

With a statistical test, Ordonez and Chen searched within a cube (of low dimension) for neighboring cells with significantly different measures [15]. The average comparison test detected cells that had very different values while they were in the same area in the cube. Searching all cell pairs with different values is a highly combinatorial problem; the authors suggest limiting the search to a reduced, user-defined space. The approach is developed with a set of SQL queries that transform the search space into a lattice. The algorithm then goes through the lattice and for each candidate cell pair performs the statistical test. A GUI allows you to view the results.

The idea proposed in this paper is promising and shows the advantage of using data mining techniques to enrich the capacities of OLAP. However, using a parametric statistical test obliges the authors to make assumptions about the data, such as for example that measure probability distribution which must follow normal distribution (a Gaussian distribution). Although the authors think that this assumption is low and generally true for the data they use, we think that it is not the case generally speaking. This theoretical assumption is a real limitation for using this approach. Furthermore, the proposed approach that highlights cells that differ greatly, is a more explanatory approach than the visualization approach.

In the context of Web 2.0 and OLAP applications, Aouiche *et al.* are particularly interested in tag clouds [1]. The principle of a tag cloud provides a visual representation of keywords on a Web site. Generally, words appear in a font size that is greater than that of the words used on the site. The keyword cloud can be viewed as a semantic abstract of the web site. Applied to OLAP, this principle can use a tag cloud to represent a cube where each keyword is a cell and where keyword size depends on the measured value of the fact (cell). This new cube representation makes it possible to use specific tag cloud operations such as sorting keywords according to their size and removing keywords with small sizes. To assist the user in his exploration of the cube, Aouiche *et al.* built classes of similar keywords and rearranged the attributes of the dimensions.

In areas other than OLAP, many works focus on how to visualize complex data. Between data mining and information retrieval, Morin worked on graphic browsing in text documents [14]. After having cleaned the data (removing stop-words, extracting lemmas or keywords,...) a lexical table (crossing documents in rows and keywords in column) was analyzed and displayed with a factor correspondence analysis.

Compared with the other approaches presented, we suggest a visualization operator (1) in the context of online analysis (2) that requires no assumptions about the data (3) that is suitable for complex objects (4) and that takes into account the semantic content of the data. Works on OLAP visualization do not deal with complex objects (even if some might be adapted to such data) and do not take into account the semantic content (only tag clouds seem to do this).

To visualize complex objects, we propose an approach that uses Correspondance Analysis, a factor analysis method known in data mining [4,13]. We use Correspondance Analysis because it makes it possible to visualize complex objects while highlighting interesting facts for analysis. This factor method is best suited to the scatter or complex object cube. When facts are complex objects, often there is no measure in the classical sense of multi-dimensional modeling. However, it is always possible to count the facts. In this case, the complex object cube with several dimensions with the COUNT function can be seen as a contingency table. Correspondence analysis can be used to display the facts.

Correspondence analysis is a method for decomposing the overall Phi-square or Inertia (proportional to the Chi-square quantity) by identifying a small number of axes in which the deviations from the expected values can be represented. Correspondance analysis produces factor axes which can be used as new dimensions, called "factor dimensions". These new axes or dimensions constitute a new space in which it is possible to plot the facts i.e. complex objects. The first factor plane (the space defined by the first two factor axes) makes it possible to display the facts. Using correspondence analysis as the visualization operator is fully justified because this method has the same goal as OLAP navigation and exploration.

Our principle may be compared to the latent semantic analysis (LSA) technique used in information retrieval [9]. Latent semantic analysis and correspondence analysis are two methods that seek to reduce the size of a space so as to highlight any links in the data. Correspondence analysis is preferred in our context because it provides more a synthetic graphic representation as well as indicators of the quality of the graphic representation.

4 Process

We provide OLAP users with a process composed of several steps:

- building the complex object cube,

– constructing the contingency table,
– completing the correspondence analysis,
– mapping complex objects on the factorial axes.

Suppose that the user wants to study keywords in order to identify the major research fields in which researchers are working. In addition, the user would like to identify researchers working on the same keywords. To meet these user requirements, we must provide the user with a visualization of publications by author and keyword, while making it possible to navigate inside the data.

4.1 Notations

According to the notations proposed in [3], let $\mathcal{C}$ be a cube with a non-empty set of d dimensions $\mathcal{D} = \{D^1, ..., D^i, ..., D^d\}$ and m measures $\mathcal{M} = \{M_1, ..., M_q, ..., M_m\}$. $\mathcal{H}^i$ is the set of hierarchical levels of dimension D^i. H^i_j is the j hierarchical level of dimension D^i. For example, the type of publication dimension D^1 has two levels: the level *Type* denoted H^1_1 and the level *Scope* denoted H^1_2.

$\mathcal{A}^{ij} = \{a^{ij}_1, ..., a^{ij}_t, ..., a^{ij}_l\}$ is the set of the l members or modalities a^{ij}_t of the hierarchical level H^i_j of the dimension D^i. The level *Scope* (H^1_2) has two members: *International*, denoted a^{12}_1 and *National*, denoted a^{12}_2.

A cell in the cube $\mathcal{C}$ is full (or empty) if it contains at least one fact (or does not contain facts).

Generally, a cube can represent a set of facts, with values involved in the measure M_q according to the members $\mathcal{A}^{ij}$ of the dimensions $\{D^1, ..., D^i, ..., D^d\}$ that characterize the facts to a given level H^i_j.

4.2 Complex Object Cube

Depending on what the user wants to analyze, a cube is defined. This constructed cube is a sub-cube from the initial cube C.

Let $\mathcal{D}'$ be a non-empty sub-set of $\mathcal{D}$ with p dimensions $\{D^1, ..., D^p\}$ ($\mathcal{D}' \subseteq \mathcal{D}$ and $p \leq d$). The p-tuple $(\Theta^1, ..., \Theta^p)$ is called a sub-cube according to $\mathcal{D}'$ if $\forall i \in \{1, ..., p\}$, $\Theta^i \neq \emptyset$ and if there is an unique $j \geq 1$ such that $\Theta^i \subseteq \mathcal{A}^{ij}$.

A sub-cube, noted $\mathcal{C}'$, corresponds to a portion from the initial cube $\mathcal{C}$. Of the d existing dimensions, only p are chosen. For each chosen dimension $D^i \in \mathcal{D}'$, a hierarchical level H^i_j is selected and a non-empty sub-set Θ^i of members is taken from all the member set $\mathcal{A}^{ij}$ of the level.

For example, the user can choose to work in the context of the publications that were written between 2007 and 2009, by authors with the status of full professor. And in this context, the user can build, as in figure 2, a cube of publications based on keywords, year of publication and the name of the first author. In our example, the sub-cube is given by $(\Theta^1, \Theta^2, \Theta^3, \Theta^4)$= ({*full professor*},{*2007, 2008, 2009*},{*Keyword 1, Keyword 2, ..., Keyword 4*},{*Author 1, Author 2, ..., Author 4*}). The measure M_q is the number of publications (*Count*).

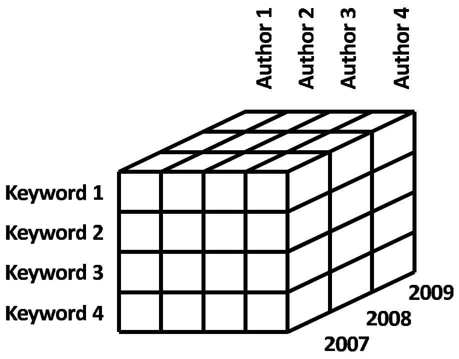


Fig. 2. Cube of publications

4.3 Contingency Table

Classically, correspondence analysis takes as input a contingency table. In our approach to online analysis of complex objects, the idea is to use traditional OLAP operators to build this contingency table.

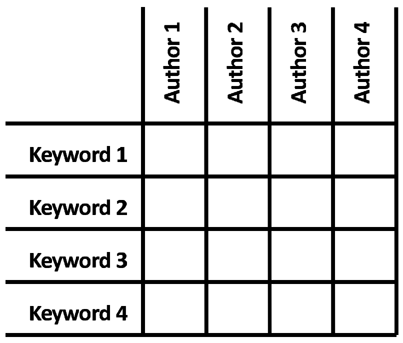
In the sub-cube $\mathcal{C}'$, the user chooses two levels (that is to say, he chooses one level for two different dimensions i and i'), on which he wants to visualize complex objects. Let Θ^i (respectively $\Theta^{i'}$) be the set of l (respectively l') members chosen for the level of the dimension i (respectively i'). The contingency table $\mathcal{T}$ has l rows and l' columns the titles of which are given by $\{a_1^{ij}, ..., a_t^{ij}, ..., a_l^{ij}\}$ and $\{a_1^{i'j'}, ..., a_{t'}^{i'j'}, ..., a_{l'}^{i'j'}\}$. At each intersection of row t and column t', are counted the facts having the members a_t^{ij} and $a_{t'}^{i'j'}$.

In our example, the contingency table crosses all the keywords with all the authors in the sub-cube. This consists in counting facts covering 3 years by doing a roll-up of the dimension *year*. This gives us a cross table with keywords in rows and authors in columns (figure 3). At the intersection of a row and a column, we have the number of publications written by an author for a given keyword. This table is ready to be processed by a correspondence analysis.

If the measure used is other than a simple count, and if it is a numerical measure, additive and with only positive values, then it is possible to use it to weigh the facts in the contingency table. The user is given the choice of using this measure as weighting or not.

4.4 Correspondence Analysis

Processing a correspondence analysis consists in projecting data on to synthetic axes so that much information is expressed by a minimum number of axes. The goal is to reduce the size of the representation space, that is to say, to reduce the number of rows and columns. Rows and columns have similar roles. The correspondence analysis makes possible simultaneous visualization of the projections of rows and columns in the same plane. The proximities between rows and columns can be interpreted.



	Author 1	Author 2	Author 3	Author 4
Keyword 1				
Keyword 2				
Keyword 3				
Keyword 4				

Fig. 3. Contingency table

In practice, the method starts by calculating the eigen values from which are deduced eigen vectors that define the factor axes. As the first two axes contain the most information, they define the first factor plane. Once row points and column points have been projected on to axes, auxiliary statistics are reported to help evaluate the quality of the axes and their interpretation. For each point, the most important statistics are the weight, the relative contribution of the point to the axis' inertia and the quality of the representation on the axis (given by the $cosine^2$). To give an interpretation of an axis and analyze proximity between points on an axis, only points which contribute strongly to the inertia of the axis (whose contribution is three times the average contribution) and which are well represented by the axis (whose $cosine^2$ is higher than 0.5) are taken into account.

In our example, publications are represented by keywords and by authors; thanks to the correspondence analysis, identifying associations between authors and keywords will be possible. Figure 4 is a very short illustration of a possible result: keyword 1 and 2 are very close, suggesting that relevant publications were written by authors working on the same topics. However, keywords 3 and 4 are far apart and therefore appear to fall under two different research fields. The position of the authors makes it possible to have an overview of the search fields in which they work. For example, authors 3 and 5 seem to work in the same fields, while they are opposed to authors 1 and 2, who themselves are opposed to author 4. We thus get a graphic summary of what authors work on, and with whom.

4.5 Visualization

The first two factor axes are retained as new factor dimensions, because the coordinates in the projected objects give their position on these new axes. They can be seen as members of dimensions. As explained above, it is possible to interpret the factor dimensions.

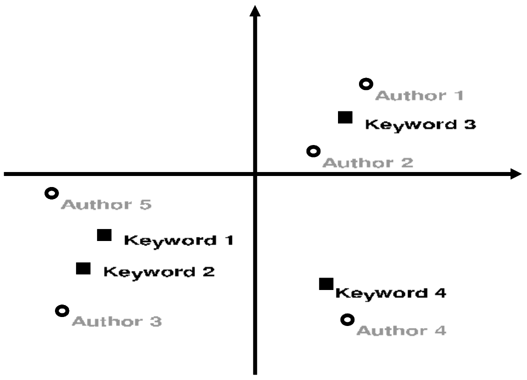


Fig. 4. First factor plane as a new graphic representation

Once the graph has been constructed (cf figure 5), an interactive tool gives, for each point, i.e. keyword or author, its statistic indicators (relative contribution and $cosine^2$). Keywords and authors that have high indicators are represented in a different color. Thus, the user sees the most relevant points for analysis. Factor analysis provides automatic help in understanding and to analyzing information. For example, the user can easily identify the most characteristic keywords, authors who work together or who do not work together and finally groups of authors working on certain keywords. In addition, if the user so requests, a photograph of the authors can replace their name. In an OLAP framework, it is efficient to use the most significant descriptors of dimensions in order to enhance the readability of the results obtained.

Furthermore, according to the OLAP principle, it is also possible on each point to perform a *drill-down* to see related publications (represented by their title). The user has another possibility of projecting a hierarchical level of another dimension into the graph. The members of this new level will be projected as points in factor space but they have not been involved in the construction of the axes. To maintain statistical consistency, only hierarchical levels whose dimensions are not in the sub-cube can be used as additional elements. A level of a dimension already used would be dependent on another level. In our example, the user could use as an additional element type of publication (journal, conference, technical report ...).

4.6 Implementation

To validate our approach, we have developed a software platform named *PUMA* (PUblication Modeling and Analysis). This prototype affords to feed the data warehouse with the publications, to build the complex object cube and to make an online analysis of publications.

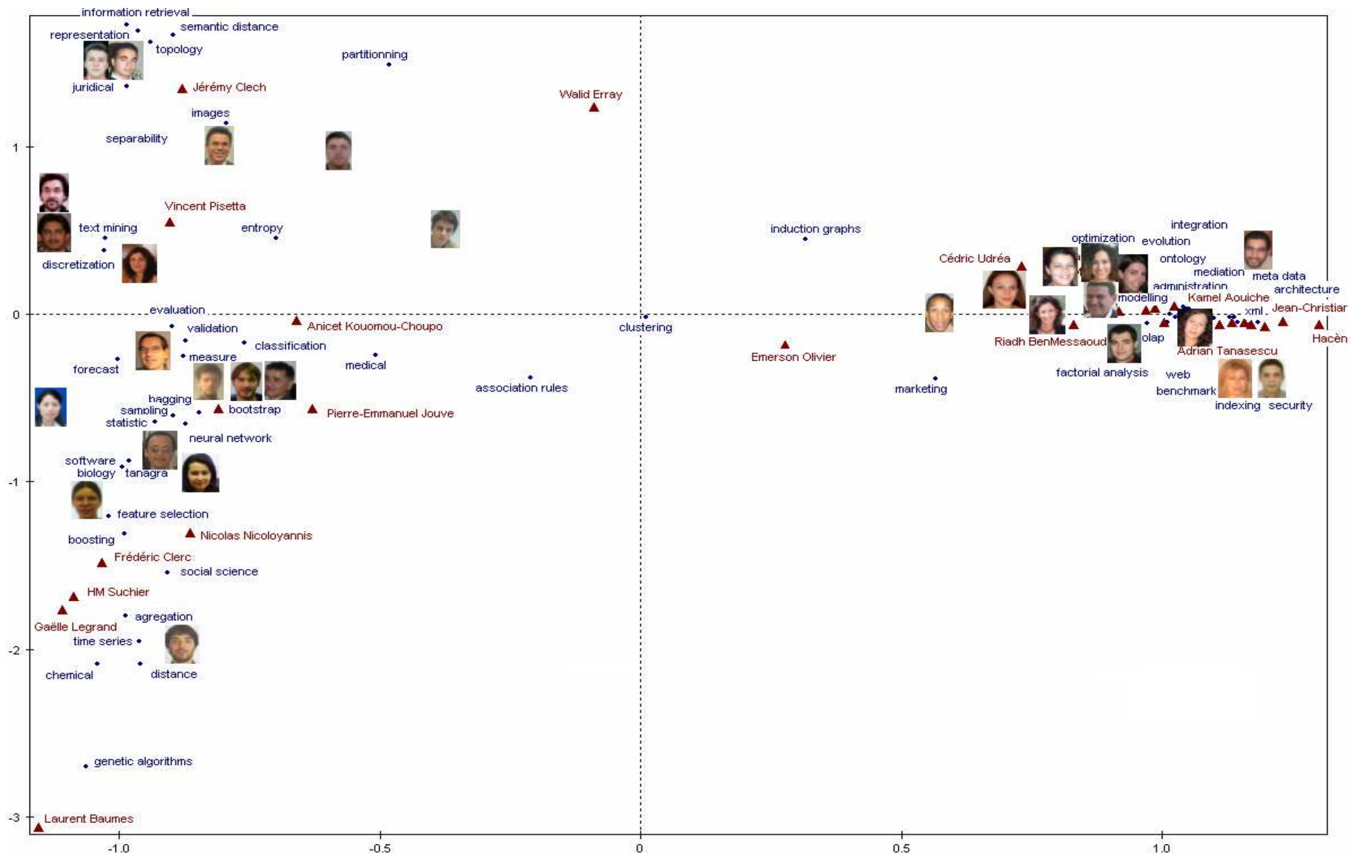


Fig. 5. Visualization of publications

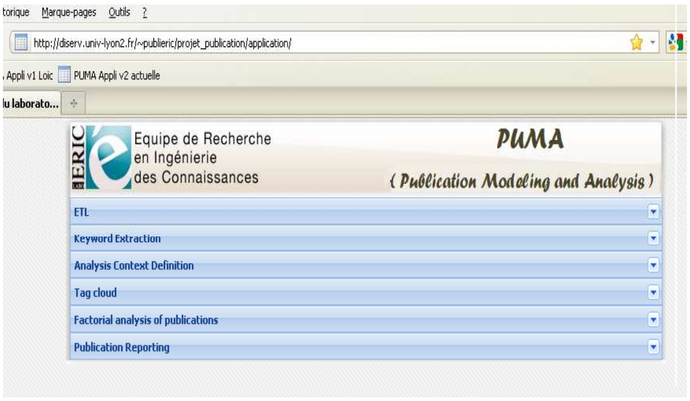

torique Marque-pages Outils ?
http://diserv.univ-lyon2.fr/~publieric/projet_publication/application/
Appli v1 Loic
PUMA Appli v2 actuelle
lu laborato...
ERIC
Equipe de Recherche en Ingénierie des Connaissances
PUMA
(Publication Modeling and Analysis)
ETL
Keyword Extraction
Analysis Context Definition
Tag cloud
Factorial analysis of publications
Publication Reporting

Fig. 6. PUMA application (PUblication Modeling and Analysis)

PUMA is a Web 2.0 Open Source application developing in *PHP*; the data warehouse is built with *MySQL*. The application also uses the software R with its package FactoMineR. It is designed as a three-third architecture with a configuration interface, an application server and a database server. From the configuration interface, the user feeds the data warehouse, defines the context of analysis by building a cube and parameters analysis he wishes to achieve. From a technical point of view, the graphic interface is managed by an *ExtJS* framework and with an *Ajax* support. From a user point of view, the application looks like an accordion that has to be deployed step by step: cube construction, correspondence analysis process, publication visualization and reporting (cf figure 6).

5 Conclusion

In this paper, we have developed an approach to online analysis for complex objects. Our approach has demonstrated the feasibility of using correspondence analysis to make it possible to visualize complex objects online. Our visualization operator takes into account the semantic content of complex objects. Our approach of displaying complex objects naturally takes its place in the online analysis. The publications case study illustrates our approach. Producing a software platform has allowed us to validate it.

In the multi-dimensional model, publications are described by keywords. Rather than asking authors to assign keywords themselves manually to their publication or rather than using an ontology, we think that it would be more relevant to automatically extract the keywords from the title, summary, or text (body) of the publication. Indeed, if the keywords were automatically extracted, they would capture some of the semantics contained in the document. Using information retrieval (IR) principles, keywords could be extracted automatically.

Furthermore, as publications contain documents and documents contain text, our idea is to use certain information retrieval (IR) techniques in order to model publications. The use of IR techniques can allow us to extract semantics from the text and this semantic information may be very helpful for modeling publications in a multi-dimensional manner. In addition to combining OLAP and data mining, the coupling of OLAP and IR should further enhance online analysis.

References

1. Aouiche, K., Lemire, D., Godin, R.: Web 2.0 OLAP: From Data Cubes to Tag Clouds. In: Cordeiro, J., Hammoudi, S., Filipe, J. (eds.) WEBIST 2008. LNBIP, vol. 18, pp. 51–64. Springer, Heidelberg (2009)
2. Ben Messaoud, R., Boussaid, O., Loudcher-Rabaseda, S.: A Data Mining-Based OLAP Aggregation of Complex Data: Application on XML Documents. International Journal of Data Warehousing and Mining 2(4), 1–26 (2006)
3. BenMessaoud, R., Boussaid, O., Loudcher-Rabaseda, S.: A multiple correspondence analysis to organize data cubes. Databases and Information Systems IV - Frontiers in Artificial Intelligence and Applications 155(1), 133–146 (2007)
4. Benzecri, J.P.: Correspondence Analysis Handbook, hardcover edition. Marcel Dekker (1992)
5. Boukraa, D., Boussaid, O., Bentayeb, F., Loudcher, S.: OLAP Operators For A Complex Object-Based Multidimensional Model. International Journal of Data Mining and Business Intelligence (DMBI), 34–46 (2010)
6. Boukraâ, D., Ben Messaoud, R., Boussaid, O.: Modeling XML warehouses for complex data: the new issues. In: Open and Novel Issues in XML Database Applications: Future Directions and Advanced Technologies, pp. 287–307 (2009)
7. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. SIGMOD Record 26(1), 65–74 (1997)
8. Codd, E.F., Codd, S.B., Salley, C.T.: Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate. Hyperion Solutions Corporation (1993)
9. Deerwester, S., Dumais, S., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by Latent Semantic Analysis. Journal of the Society for Information Science 41(6), 391–407 (1990)
10. Kimball, R.: The Data Warehouse Toolkit. John Wiley & Sons (1996)
11. Maniatis, A.S., Vassiliadis, P., Skiadopoulos, S., Vassiliou, Y.: Advanced visualization for OLAP. In: Proceedings of the 6th ACM International Workshop on Data Warehousing and OLAP (DOLAP 2003), pp. 9–16 (2003)
12. Mabit, L., Loudcher, S., Boussaid, O.: Analyse en ligne d'objets complexes avec l'analyse factorielle. In: $10^{éme}$ Conférence d'Extraction et Gestion des Connaissances (EGC 2010), pp. 381–386 (2010)
13. Greenacre, M.: Correspondence Analysis in Practice, 2nd edn. Chapman Hall, CRC (2007)
14. Morin, A.: Intensive use of correspondance analysis for information retrieval. In: Proceedings of the 26th International Conference on Information Technology Interface (ITI 2004), pp. 1–4 (2004)

15. Ordonez, C., Chen, Z.: Exploration and Visualization of OLAP Cubes with Statistical Tests. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Workshop on Visual Analytics and Knowledge Discovery, pp. 46–55 (2009)
16. Sureau, F., Bouali, F., Venturini, G.: Optimisation heuristique et génétique de visualisations 2D et 3D dans OLAP: premiers résultats. In: 5émes Journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 2009), pp. 62–75 (2009)
17. Vinnik, S., Mansmann, F.: From Analysis to Interactive Exploration: Building Visual Hierarchies from OLAP Cubes. In: Ioannidis, Y., Scholl, M.H., Schmidt, J.W., Matthes, F., Hatzopoulos, M., Böhm, K., Kemper, A., Grust, T., Böhm, C. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 496–514. Springer, Heidelberg (2006)

Knowledge Dimension in Business Process Modeling

Ligita Businska and Marite Kirikova

Faculty of Computer Science and Information Technology,
Riga Technical Univesity, Latvia
{ligita.businska,marite.kirikova}@rtu.lv

Abstract. Various business process modeling notations give an opportunity to include elements that belong to different enterprise architecture models in the business process representation. Each model that relates to the business process via its elements can be viewed as a dimension of the business process. Thus the organizational structure model (performer model), goal model, data model, location model, and other models represent a particular dimension of the business process. One of the dimensions that have not yet evolved into a model, which could be easily related to the business process, is knowledge dimension. The paper presents knowledge state transition model rooted into the notion of knowledge code, analyses knowledge representation capabilities of existing business process modeling languages, and proposes a business process activity template, which includes internal and external representations of knowledge dimension. The template helps to clarify several issues with respect to knowledge dimension of business process models and to move forward towards the business process modeling language that can incorporate all modes of knowledge included in knowledge state transition model.

Keywords: data, information, knowledge, business process model.

1 Introduction

The period of distrust in business process model based approaches due to unsuccessful re-engineering efforts, which took place in the previous century, is over; and business process modeling again becomes an important topic in scientific literature [1-4]. However, it is worth to remember that business process engineering has to be a holistic approach and take into consideration various aspects or dimensions of the business system, including organizational and individual knowledge [1-5].

While importance of knowledge dimension is well recognized, there is no clear theoretical background and successful practical experiments of inclusion of this dimension in business process modeling languages. In such languages as IDEF0, IDEF3, EPC diagrams in ARIS tool, GRAPES BM in GRADE tool, UML 2.0 activity diagram, and BPMN 2.0 data, information and material flows are often represented by the same symbols and without any unambiguous definitions of the concepts. On the other hand, knowledge modeling languages (KMDL, GPO-WM, PROMOTE, and RAD) allow to model knowledge, but do not address process logic to full extent and thus there is no possibility to represent data [6, 7]. Currently, from the point of view of various ways how data, information, and knowledge are used in organizations, the following issues are not yet fully supported in any of the above-mentioned business process modeling and knowledge modeling languages:

- Possibility to separate information and data during business process modeling
- Opportunity to identify the owner of data, information, and knowledge
- Possibility to identify, plan, and manage knowledge of the role required for participating in a particular activity and linking this knowledge to the organizational competence model
- Possibility to evaluate the amount of lost organizational knowledge if a person – owner of knowledge – leaves the organization, i.e., to identify which tacit knowledge in which cases should be transformed into explicit knowledge, such as documents, rules, systems, etc.
- Opportunity to improve understanding about the knowledge usefulness, validity, and relevance for particular activities in a process
- Opportunity to enable competence requirements management and proactive training based on a process reengineering impact analysis

The goal of this paper is to provide theoretically sound representation of the knowledge dimension that would give an opportunity to support above-mentioned issues in business process modeling. We aim at obtaining new knowledge helpful for developing modeling languages that could handle all relevant aspects related to knowledge dimension.

We have already tried to address knowledge dimension by using BPMN notation in our previous work [6]. This lead to the introduction of specific symbols for data, information and knowledge objects. Experiments with the notation revealed that the relationship between the phenomena behind the symbols is somewhat unclear in the modeling process. Therefore, in this paper, we focus on analysis of this relationship by investigating intersection of modern information theory assumptions and knowledge management definitions of information and knowledge.

In Section 2, we ponder over the terms data, information, and knowledge and come to the conclusion that the use of information codes as a supplementary term helps to clarify relationship between previous three terms. The state transition model of knowledge, which is taken as a theoretical basis for inclusion of knowledge dimension in business process model, is represented in this section, too. We use all four terms (data, information, knowledge, and information code) to define information interaction in homogenous and heterogeneous environments. In Section 3, we analyze existing business process modeling languages in the context of information interaction. In Section 4, the template of a business process activity with visible knowledge dimension is proposed and an example of its use is represented. In Section 5, pros and cons of the proposed approach are discussed.

2 Constituents of Knowledge Dimension

Data, information, and knowledge are concepts that are widely used in various fields of human activities. Their meaning is discussed in various fields of research since ancient times. Despite of numerous research works and scientific theories on interpretation of data, information, and knowledge in psychology, epistemology, social science, philosophy, cognitive science, and information theory; these terms are still used intuitively and often lack explicit unified definition within the areas of research. Most often data are associated to databases and knowledge is related to human beings, while information is attributed to both – databases and human beings. Uncertainty exists not only in definitions, but also in the practical use of the concepts. Usually in representation of flows between activities in existing languages of business process modeling do not distinguish between definitions of data, information, and knowledge and do not provide specific symbols for their representation in business process model [6]. Let us consider a business case when a seller who works with a cash register perceives barcodes and the data on the receipts just as data without certain meaning. However, for a commodity researcher this data provides meaningful information on goods; and processing of this information brings knowledge about sales volumes. In another case, a bank employee uses several information systems (IS) with different level of intelligence: without data interpretation, with data interpretation, and with ability to create new knowledge. Here it should be possible to distinguish between different types of inputs and outputs according to the level of intelligence of the system and to identify whether the bank employee has knowledge that enables him/her to understand and interpret data provided by the system. One more case that shows the difference between information and knowledge flows is situation where bank employee increases knowledge and speeds up client servicing when he/she tries to remember which products were corresponding to client goals; but this information about the goals of a particular client becomes insignificant when the next client comes to the desk. Thus, information flow appears when the bank employee remembers it temporarily and does not transform it into knowledge. Alternatively, the knowledge flow is significant if the bank employee accumulates information in order to improve his/her decision-making ability. From above-mentioned cases, we can see that it is essential to understand what exactly (data, information, or knowledge) is transferred between the activities and how this could be represented separately in the process models.

In this paper, we do not discuss various interpretations of the above-mentioned terms deeply [8, 9, 10, 11, 12]. We focus on the relationship between data, information, and knowledge and rely upon the following observations and assumptions:

- Knowledge is located in the knowledge holder (natural or artificial)
- Knowledge in the knowledge holder (e.g., human brain) has a particular structure that may be regarded as a “mental model”. The “mental model” can be natural or artificial, tacit and externalized, implicit and explicit
- Any business process involves a knowledge process which is performed by a natural or artificial knowledge holder
- If several knowledge holders are involved in the business process - data, information and knowledge exchange between them is possible. This exchange differs from the exchange of other substances as it is asymmetric: The amount of given information may differ from the amount of received information; and the knowledge holder does not lose knowledge by giving information based on this knowledge.

To obtain a holistic and at least semi-formal view of the relationship between data, information, and knowledge we use theory that shows that in information exchange a substance called ‘information codes’ is involved [8], i.e., information exchange is accomplished via information codes.

Suppose the knowledge holder (object O_1 provides information codes T_1 to another knowledge holder (object O_2). The state transition in O_2, which receives this information, is illustrated in Fig. 1. In the first phase, the object O_2 receives particular information code Ic_1. To perceive the code the object needs a particular “linguistic device” that can recognize the code (e.g., if the code is information in English, it can be recognized if there is a “device” that can handle English). The received code is transformed into data Δd. Thus, *data are functional values of information codes*, which correspond to new parameters of object state obtained in interaction with another object.

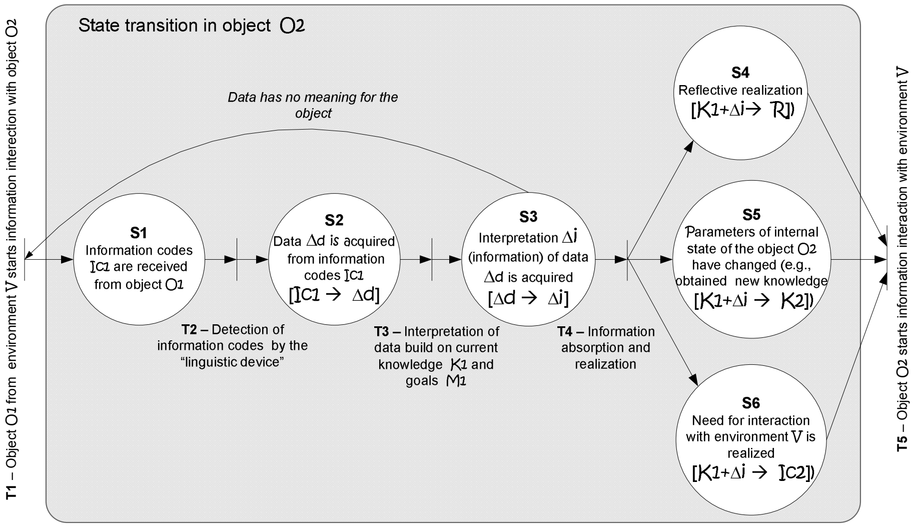

T1 – Object O1 from environment V starts information interection with object O2
State transition in object O2
Data has no meaning for the object
S1
Information codes Ic1 are received from object O1
S2
Data Δd is acquired from information codes Ic1
[Ic1 → Δd]
S3
Interpretation Δi (information) of data Δd is acquired
[Δd → Δi]
S4
Reflective realization
[K1+Δi→ R])
S5
Parameters of internal state of the object O2 have changed (e.g., obtained new knowledge
[K1+Δi → K2])
S6
Need for interaction with environment V is realized
[K1+Δi → Ic2])
T2 – Detection of information codes by the “linguistic device”
T3 – Interpretation of data build on current knowledge K1 and goals M1
T4 – Information absorption and realization
T5 – Object O2 starts information interaction with environment V

Fig. 1. State transition in the knowledge owner when it receives information codes

In the next phase, object O_2 defines the meaning of obtained data Δd. This is a subjective interpretation of Δi by current knowledge of K_1 of O_2 taking into consideration M_1 – the set of its current needs or goals. According to [9] structured and processed data is information that is time dependent (relevant only in a given point of time) and correct with respect to the processed data set. In general, the amount of received information can be calculated as a difference between knowledge obtained after data interpretation and knowledge possessed before the interaction with object O_1: $\Delta i = Z_1 - Z_2$. It can be regarded as a measure of reduction of uncertainty for choosing actions in order to achieve particular goals M_1 [13].

Information exists from the moment the data is interpreted until the moment when the information has been absorbed or included in the mental model of the object. The content or structure of the mental model (including procedural and declarative knowledge, which is stored in it) can be changed as the result of information absorption.

Finally, recognition of obtained information Δi takes place. The implementation can lead to changes of internal state parameters of object O_2 or/and to the next cycle of interaction with the environment. Several overlapping options of implementation can take place: (1) a reflective action: $K_1+\Delta i \rightarrow R$; (2) object O_2 delivers appropriate set of information codes: $K_1+\Delta i \rightarrow Ic_2$ in case of starting the next cycle of interaction with object(-s) from its environment; (3) object's mental model (internal state) can change under certain conditions when obtaining new knowledge: $K_1+\Delta i \rightarrow K_2$. According to [14] *knowledge is reasoning about data* that is stored in object's "mental model" in order to promote action, problem solving, decision-making, learning, and teaching. Knowledge is a higher organizational level of data that allows their specific interpretation. Requirements to data organization level can vary from a simple grouping of the data to complicated data hyper-structures.

Thus according to [8] a single cycle of information interaction between object and its environment has three sequential phases: (1) the object receives information codes from its environment, (2) it interprets the obtained codes, and finally (3) it recognizes the information obtained by interpretation (reflects upon it, absorbes it, and/or puts it into the action). In Fig. 2 a simplified example with two objects (the process performer (analyst) and the document that includes interview protocols) is shown.

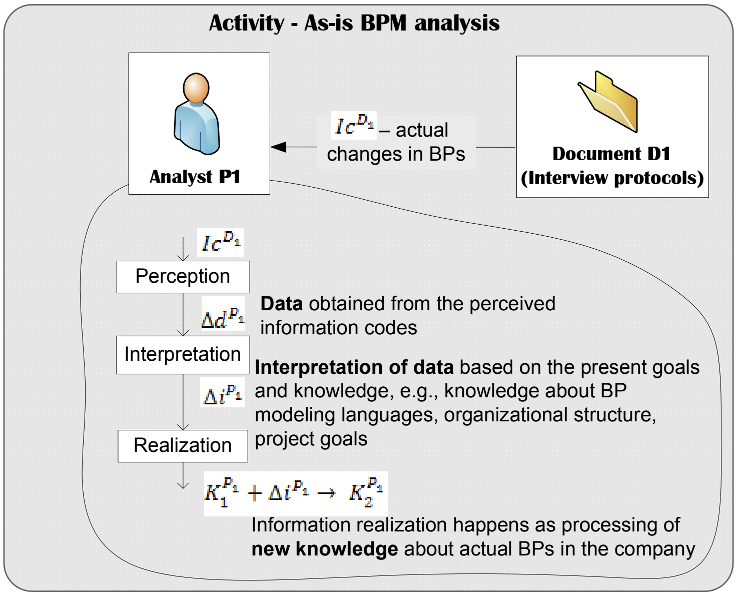

Activity - As-is BPM analysis
Analyst P1
Ic^{D_1} – actual changes in BPs
Document D1 (Interview protocols)
Ic^{D_1}
Perception
Δd^{P_1}
Data obtained from the perceived information codes
Interpretation
Δi^{P_1}
Interpretation of data based on the present goals and knowledge, e.g., knowledge about BP modeling languages, organizational structure, project goals
Realization
$K_1^{P_1} + \Delta i^{P_1} \rightarrow K_2^{P_1}$
Information realization happens as processing of new knowledge about actual BPs in the company

Fig. 2. A simplified example of an activity

In the above-given example the analyst performs the activity of analyzing as-is business process model that is described in Document D1. When reading this document, the analyst perceives information codes by his/her receptors and obtains certain set of data. In this stage, perceived data does not have any meaning for the analyst. According to above-mentioned information theory, perceived data are compared with the present goals/needs and background knowledge (e.g., PB modeling languages, organizational structure, project goals) of the analyst. As the result the subjective interpretation of data (or information) is obtained. Finally, the information is recognized by the analyst as new knowledge about actual business processes in the company. It means that the recognition process changes the mental model of the analyst (e.g., by enriching it with new links or nodes, reorganizing or generalizing existing structure, or adding new values for structure elements).

The performer of a business process can receive information codes in three different ways, namely, from human, from active artificial object, and from passive artificial object. Depending on the situation the interchange of information codes can take place in homogenous (human-human, IS-IS) or heterogeneous (human-IS, IS-document, human-document) environments. In Fig. 3 and 4 internal changes in objects (knowledge holders) are illustrated. Fig. 3.A shows information code interchange and new knowledge (natural or artificial) development in homogenous environment (on the left: human-human and on the right: computer IS-IS). Fig. 3.B illustrates how natural or artificial knowledge holder interacts with the passive knowledge holder (document).

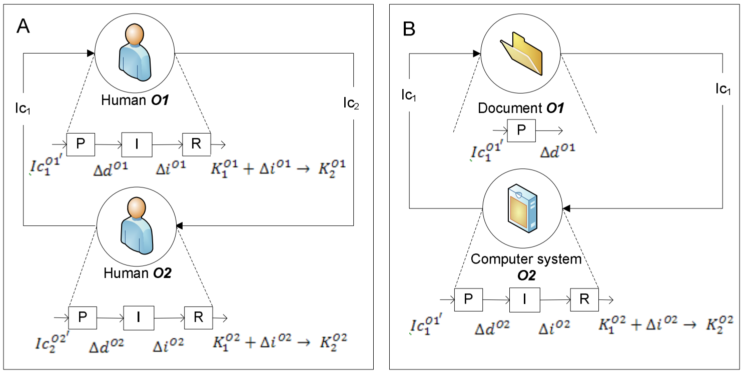


Fig. 3. Different types of information interaction. **A:** Information interaction in homogenous environments; **B:** Information interaction between active knowledge holders and passive knowledgeholders (P – perception, I – interpretation, R – recognition)

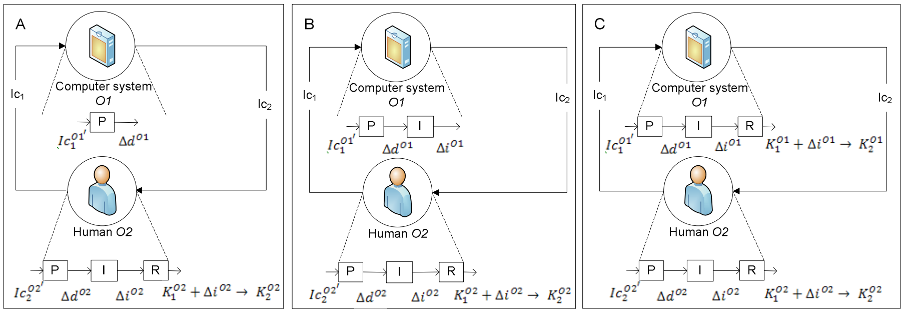


Fig. 4. Information interaction in heterogeneous environment (among active objects) (P – perception, I – interpretation, R – recognition)

Figure 4 illustrates heterogeneous environment with two different types of knowledge holders. The interchange and knowledge development can proceed differently depending on the level of intelligence of the IS (computer system): from the left to right: without data interpretation; with data interpretation only, and with learning ability.

The above analysis of information interaction shows that changes in knowledge are initiated by perception of particular information codes. Thus, for representing the knowledge dimension it would be necessary to show knowledge before and after perception of information codes as well as the coded information itself. The potential of contemporary business process modeling languages in this regard is examined in the next section.

3 Information Exchange in Business Process Context

In our previous work [6] we analyzed different attempts to include knowledge dimension in business process modeling and knowledge modeling languages and we proposed to integrate knowledge-oriented modeling language KMDL [15] and BPMN notation [15]. In this work three different objects were used: knowledge objects, information objects and data objects. However, further experiments with the integrated notation showed that it is difficult to distinguish between data and information objects. Theoretical issues discussed in the previous section clarify the reason behind this difficulty. It shows that data is an internal rather than the external phenomenon with respect to the knowledge holder; and interchange of perceivable knowledge is accomplished via information codes. None of the approaches analyzed in [6 and 7] took into consideration information codes and therefore they are not directly applicable for representation of knowledge dimension in the way it is described in the previous section. In particular, BPMN provides opportunity to model only information and data flow using the same modeling constructs [6, 15].

In the proposed approach, the main difference from existing notations is outlining the owner of the data, information, and knowledge, which can be a human or an artificial object. The processing of information codes occurs inside the actor, but in the model it is shown just as the result of the processing: obtained knowledge or received interpretation of data (information). Additionally, actors have a link to materialized or non-materialized knowledge, which is used as a resource for processing information codes.

Both business process modeling [6] and knowledge modeling [7] approaches concern the linkage between the business processes and knowledge. Since in practice, knowledge modeling languages are used less often than business process modeling languages; in this paper we mainly consider business process modeling languages in order to see how appropriate they are for inclusion of knowledge dimension. The following business process modeling languages were analyzed: GRAPES BM – in GRADE tool [17], EPC diagrams in ARIS [18], KMDL [19], IDEF 0 [20], UML 2.0 activity graphs [21], and BPMN 2.0 [16]. The languages were analyzed from the following two points of view (1) possibilities to represent data and knowledge (Table 1) and (2) possibilities to represent process logics (Table 2). Both views are important for representation of static and dynamic aspects of knowledge in individual knowledge holders and in the whole business process.

The contents of Table 1 and Table 2 are based on the results obtained in the previous research [6, 7]. In [6 and 7] we analyzed syntaxes and semantic of modeling languages in order to understand their data, information, knowledge interpretation and modeling capabilities. In addition, we illustrated the described differences of data, information, and knowledge representation by modeling the same business process in the selected modeling languages.

Table 1. Representation of inputs, outputs, and resources ("-" means "does not support"; "-/+" means "somewhat supports"; "+" means "inclusion is possible"; "++" means "almost fully supports", and "+++" means "supports fully")

Criteria	*GRAPES*	*EPC*	*IDEF0*	*KMDL*	*UML*	*BPMN*
Input/output [data]	+	+++	+	-	+	++
Input/output [information]	+	+++	+	+	+	++
Input/output [knowledge]	-	+/-	-	+++	-	-
Resource [knowledge]	-	-	-	-	-	-
Resource [human]	+	++	+	-	-	+
Resource [artificial]	+	+	+	-	-	+
Resource [data store]	+	+	-	-	+	+

Table 2. Representation of process logics ("-" means "does not support"; "-/+" means "somewhat supports"; "+" means "inclusion is possible"; "++" means "almost fully supports", and "+++" means "supports fully")

Criteria	*GRAPES*	*EPC*	*IDEF0*	*KMDL*	*UML*	*BPMN*
Process management	-/+	-/+	+	-	-/+	-/+
Controls	-/+	-/+	+	-	-/+	-/+
Decision points	+	+	-	+	+	+
Control flows	+	++	-	+	++	+++
Events	+	++	-	-	+	+++

We can conclude that BPMN and ARIS EPC are more expressive for modeling process logic, decision points and control flows than other languages, while BPMN offers extended notation for control flow organization; and ARIS EPC is the most expressive in linkage of modeling dimensions (e.g., IS, products, organization, risks, and key performance indicators). Knowledge inclusion into the model is possible in KMDL and ARIS EPC, in the most convenient way data can be represented by UML activity diagram, and, for information flows representation, BPMN collaboration and choreography models can be used.

In Figure 5 a schematic comparison of above-mentioned notations and languages is given according to five business process modeling dimensions, namely: data, information, knowledge, material, and process. The diagrams show potential capabilities of the discussed notations or languages to represent particular business process modeling dimension. Results are presented in the interval from 0 to 4, where 0 means that the notation does not provide any symbols for the particular dimension and 4 means that the notation has a considerable number of symbols for the particular dimension.

From the point of view of knowledge dimension representation, the least feasible is IDEF0. Obviously, this language has to be modified in case it is taken as a basis for the representation of knowledge dimension. However, IDEF0 allows distinguishing between different kinds of the input flows (controls, inputs, and resources) that are important for extending process model with knowledge dimension. Therefore, it makes sense to examine a possibility to construct extended IDEF0 based form of process representation to see how appropriate the notation is for incorporating the knowledge dimension. IDEF0 based approach presented in this paper is one of the alternatives how to integrate data, information, and knowledge representation in the process model; and additional future research and experiments with other notations are needed to select the most appropriate modeling technique.

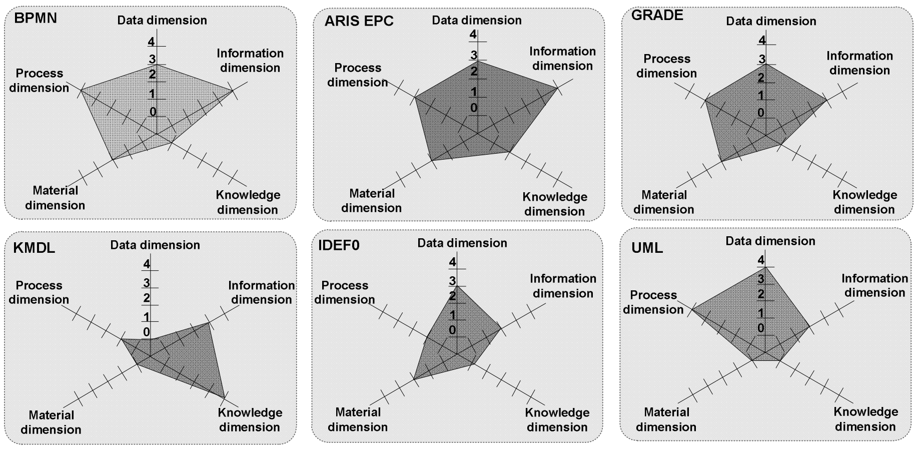


Fig. 5. Schematic comparison of process-oriented and knowledge-oriented languages and notations

4 Transparent Representation of Knowledge Dimension

In this section, we propose one of the approaches of representing business process activities with knowledge dimension. We strive to show the proposed ideas graphically. The representation is based on IDEF0 notation. Its purpose is to experimentally examine the applicability of IDEF0 for inclusion of knowledge dimension in business process model; and it should not yet be considered as a new business process modeling notation. IDEF0 was chosen as the basis for activity template, because it gives an opportunity to distinguish between controls (relates to knowledge holder's goals (see Section 2)), inputs/outputs (received and produced information codes), and resources (knowledge in the holder). However, since IDEF0 notation is weak in representing logic of the process, in our further research, we intend to combine it with other notations that give more tools for control and decision points modeling. The activity template and example of its use are represented in Fig. 6, 7 and 8.

Each *Activity* (Fig. 6.A) corresponds to one of different combinations of interaction between human, computer systems, and documents as shown in Fig. 3-4. Social processes among performers inside the activity are not represented (Fig. 6.B). The activity template has the following attributes: *Activity name*, *Performers* of the activity (human or artificial (computer) system). For knowledge intensive activities there is an additional attribute *Type* with possible values *Socialization*, *Externalization*, *Combination*, and *Internalization*. Visually, these attributes and their values are positioned in the central part of the template. The central part is surrounded by four blocks that correspond to four types of knowledge, namely: control knowledge *Kc*, input knowledge *Ki*, output knowledge *Ko*, and resource knowledge *Kr*. This is knowledge that is inside the knowledge holders (natural and/or artificial) participating in the activity and can be referred to as tacit knowledge. Each block of the tacit knowledge can be linked to particular artifacts: input artifacts *I*, output artifacts *O*, resource artifacts *R*, and control artifacts *C,* which in essence are information codes perceived by tacit (natural or artificial) knowledge of the performers of the process. Each block *Kc*, *Ki*, Ko, and *Kr*, of the template can be related to particular concepts of the representation of organizational "mental model", if such is maintained.

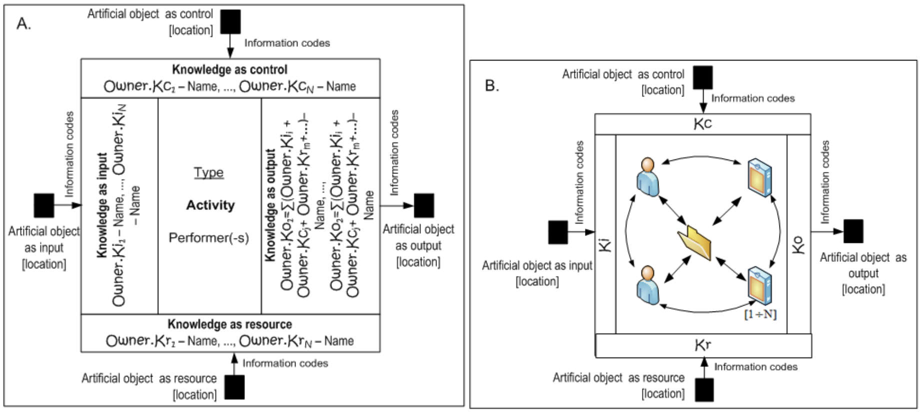

A.
Artificial object as control [location]
Information codes
Knowledge as control
Owner.Kc1 – Name, ..., Owner.KcN – Name
Knowledge as input
Owner.Ki1 – Name, ..., Owner.KiN – Name
Type
Activity
Performer(-s)
Knowledge as output
Owner.Ko1=∑(Owner.Ki + Owner.Kcj+ Owner.Krm+...)– Name, ...,
Owner.Ko2=∑(Owner.Ki + Owner.Kcj+ Owner.Krm+...)– Name
Artificial object as input [location]
Artificial object as output [location]
Knowledge as resource
Owner.Kr1 – Name, ..., Owner.KrN – Name
Artificial object as resource [location]
B.
Kc
Ki
Ko
Kr
[1÷N]

Fig. 6. Activity with a knowledge dimension: **A:** activity template; **B:** activity zoomed in (this information is not presented in the template)

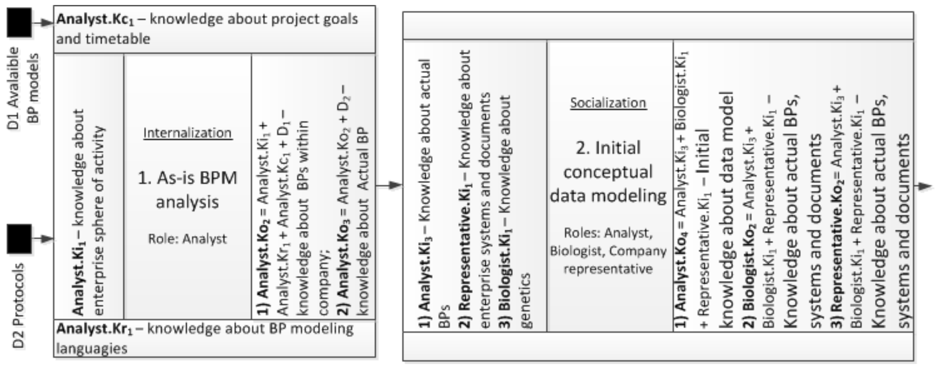

D1 Avalaible BP models
D2 Protocols
Analyst.Kc1 – knowledge about project goals and timetable
Analyst.Ki1 – knowledge about enterprise sphere of activity
Internalization
1. As-is BPM analysis
Role: Analyst
1) Analyst.Ko2 = Analyst.Ki1 + Analyst.Kr1 + Analyst.Kc1 + D1 – knowledge about BPs within company;
2) Analyst.Ko3 = Analyst.Ko2 + D2 – knowledge about Actual BP
Analyst.Kr1 – knowledge about BP modeling languagies
1) Analyst.Ki3 – Knowledge about actual BPs
2) Representative.Ki1 – Knowledge about enterprise systems and documents
3) Biologist.Ki1 – Knowledge about genetics
Socialization
2. Initial conceptual data modeling
Roles: Analyst, Biologist, Company representative
1) Analyst.Ko4 = Analyst.Ki3 + Biologist.Ki1 + Representative.Ki1 – Initial knowledge about data model
2) Biologist.Ko2 = Analyst.Ki3 + Biologist.Ki1 + Representative.Ki1 – Knowledge about actual BPs, systems and documents
3) Representative.Ko2 = Analyst.Ki3 + Biologist.Ki1 + Representative.Ki1 – Knowledge about actual BPs, systems and documents

Fig. 7. Fragment of the business process model *Development of logical data model of Bioinformatics Company* represented by the template

Table 3. Four types of knowledge: input, resource, control, and output

1. As-is BPM analysis	2. Initial conceptual data modeling
INPUT – Analyst's knowledge about the enterprise domain of activity **(Analyst.Ki_1)** **RESOURCE** – Analyst's knowledge about BP modeling languages (**Analyst.Kr_1**) **CONTROL** – Analyst's knowledge about the project goals and timetable (**Analyst.Kc_1**) **OUTPUT** – 1) Analyst's knowledge about BPs within company is obtained based on the combination of Input, Resource, and Control knowledge and knowledge embedded in Document 1 (**Analyst.Ko_2 = Analyst.Ki_1 + Analyst.Kr_1 + Analyst.Kc_1 + D_1**) 2) Analyst's knowledge about actual BPs is obtained based on the combination of knowledge about BPs and knowledge embedded in Document 2 (**Analyst.Ko_3 = Analyst.Ko_2 + D_2**)	**INPUT** – 1) Analyst's knowledge about actual BPs obtained in the previous activity (**Analyst.Ki_3**) 2) Company representatives knowledge about the enterprise systems and documents (**Representative.Ki_1**) 3) Biologist knowledge about genetics (**Biologist.Ki_1**) **OUTPUT** – 1) Analyst's initial knowledge about the data model obtained during socialization activity (**Analyst.Ko_4 = Analyst.Ki_3 + Biologist.Ki_1 + Representative.Ki_4**) 2) Biologist's knowledge about actual BPs, systems and documents (obtained during socialization activity) (**Biologist.Ko_2 = Analyst.Ki_3 + Biologist.Ki_1 + Representative.Ki_4**) 3) Company representatives knowledge about actual BPs, systems and documents (obtained during socialization activity) **(Representative.Ko_2 = Analyst.Ki_3 + Biologist.Ki_1 + Representative.Ki_4**)

To illustrate the proposed template the fragment of a logical data model of Bioinformatics Company is illustrated (Fig. 7 and 8). The given fragment consists of two consecutive activities performed to create a logical data model of the Company. First, the analyst studies available documents. During this activity he/she perceives information codes that are embodied in the documents. Then the analyst together with the biologist and the company representative discusses knowledge obtained in the previous activity. This way they exchange information codes and each of them processes perceived codes inside his/her brains according to personal mental model and goals. As a result, they could extend their mental models with new data or, under certain conditions, obtain new knowledge. Figure 7 represents the business process model using the proposed activity template (see. Fig. 6.A). Additionally, Table 3 describes inputs, resources, controls, and outputs of the activities in detail.

Since currently our aim is not to establish a new business process notation, the main effort is put in reflecting knowledge dimension graphically. More studies and experiments are needed to determine the best way of the reflection. The main limitation of the proposed graphical representation is that it looks complex, especially, if the number of activities in the process model is increased. One of the solutions for reducing complexity of comprehension is developing appropriate modeling tool with embodied functionality to switch between different views of the process, e.g., process view and data/information/knowledge view. Pros and cons of the proposed approach are analyzed in Section 5 in more detail.

5 Discussion of Proposed Approach and Conclusions

To evaluate the proposed graphical notation, we consider the business process example in Fig. 8 and its representation using the proposed template (Fig. 9).

By comparing models in Fig. 8 and 9, it is possible to make some preliminary conclusions that are summarized in the Table 4. The table shows that the proposed approach opens a possibility to consider issues of organizational knowledge that cannot be included in conventional BPMN models. Nevertheless, the approach has several drawbacks related to representation of data stores, events, control flows, and decision points.

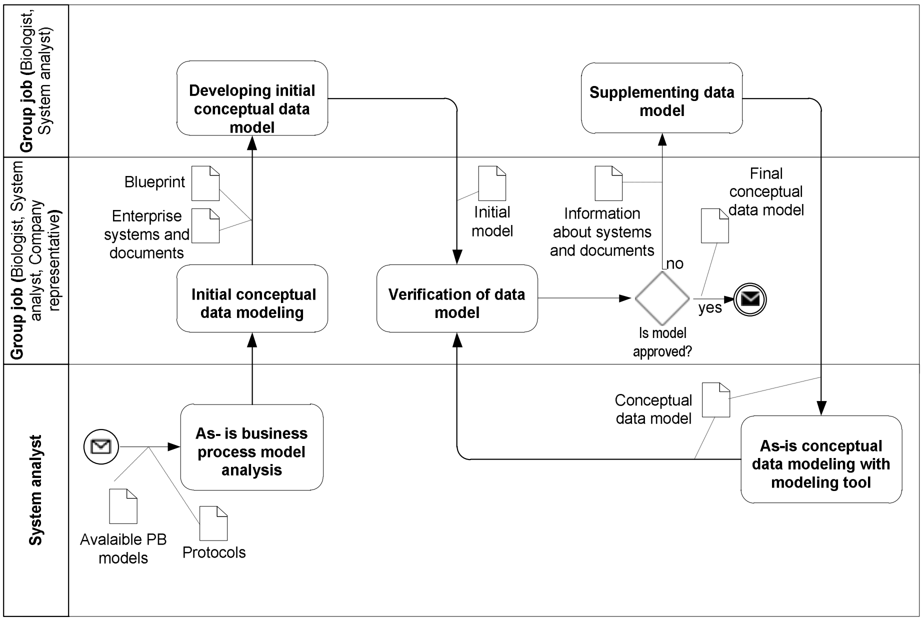
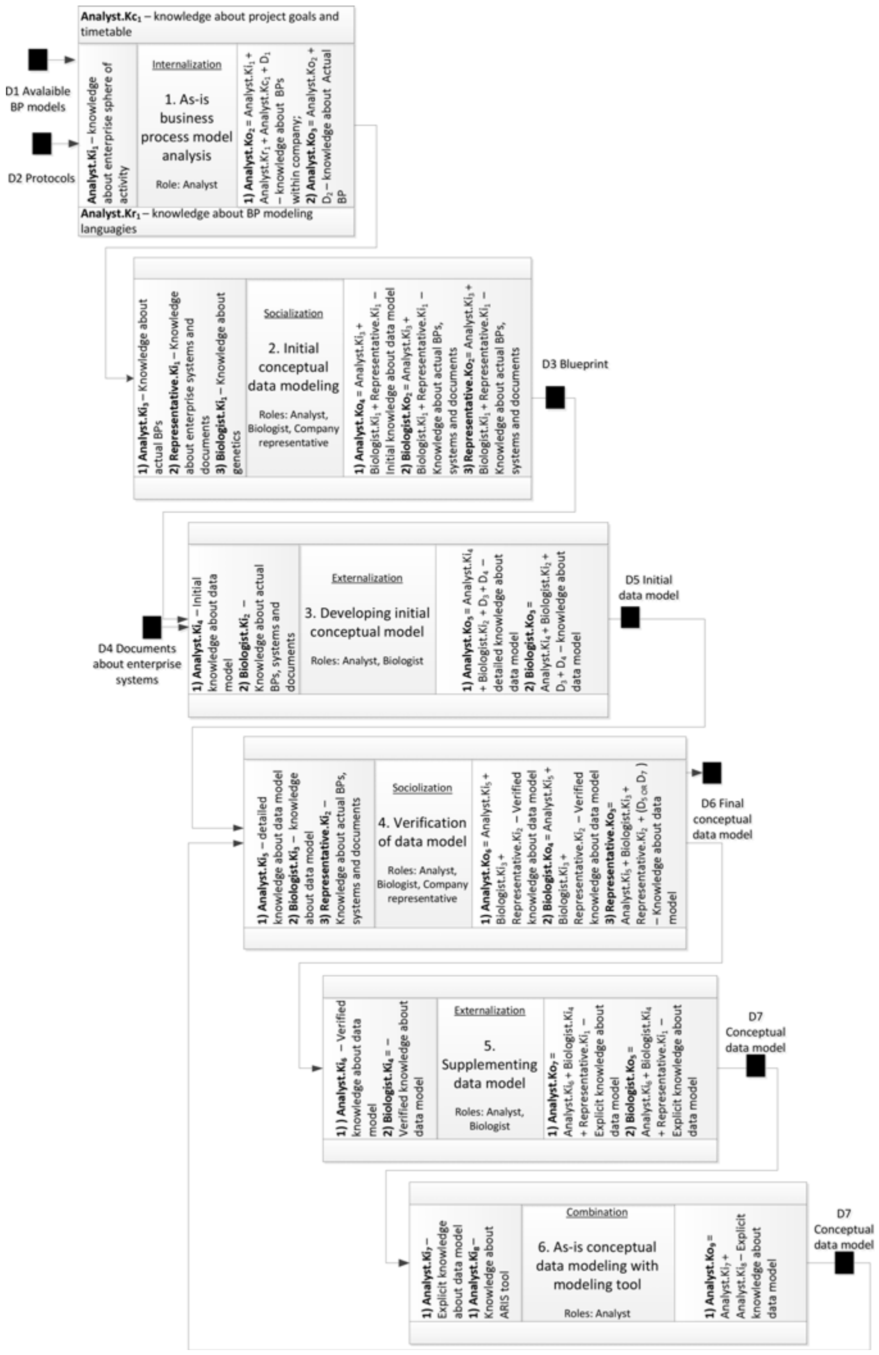

Group job (Biologist, System analyst)
Group job (Biologist, System analyst, Company representative)
System analyst
Developing initial conceptual data model
Supplementing data model
Blueprint
Enterprise systems and documents
Initial model
Information about systems and documents
Final conceptual data model
no
Initial conceptual data modeling
Verification of data model
yes
Is model approved?
Conceptual data model
As- is business process model analysis
As-is conceptual data modeling with modeling tool
Avalaible PB models
Protocols

Fig. 8. Business process model developed in BPMN

In business process analysis, design, engineering, and reengineering, it is important to have a holistic view of the enterprise. Since organizational knowledge is an essential aspect of an enterprise, there is a need of transparent linkage between the business process model and organizational and individual knowledge. In order to achieve this transparency the paper proposes a new activity template that gives visual means to relate business process to organizational knowledge and to analyze knowledge circulation in a business process. The model presented in the paper is in its experimental stage. As discussed in Sections 4 and 5, it helps to deal with the following issues related to knowledge dimension of business process models:



Analyst.Kc₁ – knowledge about project goals and timetable
D1 Avalaible BP models
D2 Protocols
Analyst.Ki₁ – knowledge about enterprise sphere of activity
Internalization
1. As-is business process model analysis
Role: Analyst
1) Analyst.Ko₂ = Analyst.Ki₁ + Analyst.Kr₁ + Analyst.Kc₁ + D₁ – knowledge about BPs within company;
2) Analyst.Ko₃ = Analyst.Ko₂ + D₂ – knowledge about Actual BP
Analyst.Kr₁ – knowledge about BP modeling languagies
1) Analyst.Ki₃ – Knowledge about actual BPs
2) Representative.Ki₁ – Knowledge about enterprise systems and documents
3) Biologist.Ki₁ – Knowledge about genetics
Socialization
2. Initial conceptual data modeling
Roles: Analyst, Biologist, Company representative
1) Analyst.Ko₄ = Analyst.Ki₃ + Biologist.Ki₁ + Representative.Ki₁ – Initial knowledge about data model
2) Biologist.Ko₂ = Analyst.Ki₃ + Biologist.Ki₁ + Representative.Ki₁ – Knowledge about actual BPs, systems and documents
3) Representative.Ko₂= Analyst.Ki₃ + Biologist.Ki₁ + Representative.Ki₁ – Knowledge about actual BPs, systems and documents
D3 Blueprint
D4 Documents about enterprise systems
1) Analyst.Ki₄ – Initial knowledge about data model
2) Biologist.Ki₂ – Knowledge about actual BPs, systems and documents
Externalization
3. Developing initial conceptual model
Roles: Analyst, Biologist
1) Analyst.Ko₅ = Analyst.Ki₄ + Biologist.Ki₂ + D₃ + D₄ – detailed knowledge about data model
2) Biologist.Ko₃ = Analyst.Ki₄ + Biologist.Ki₂ + D₃ + D₄ – knowledge about data model
D5 Initial data model
1) Analyst.Ki₅ – detailed knowledge about data model
2) Biologist.Ki₃ – knowledge about data model
3) Representative.Ki₂ – Knowledge about actual BPs, systems and documents
Sociolization
4. Verification of data model
Roles: Analyst, Biologist, Company representative
1) Analyst.Ko₆ = Analyst.Ki₅ + Biologist.Ki₃ + Representative.Ki₂ – Verified knowledge about data model
2) Biologist.Ko₄ = Analyst.Ki₅ + Biologist.Ki₃ + Representative.Ki₂ – Verified knowledge about data model
3) Representative.Ko₃= Analyst.Ki₅ + Biologist.Ki₃ + Representative.Ki₂ + (D₅ or D₇) – Knowledge about data model
D6 Final conceptual data model
1)) Analyst.Ki₆ – Verified knowledge about data model
2) Biologist.Ki₄ = – Verified knowledge about data model
Externalization
5. Supplementing data model
Roles: Analyst, Biologist
1) Analyst.Ko₇ = Analyst.Ki₆ + Biologist.Ki₄ + Representative.Ki₁ – Explicit knowledge about data model
2) Biologist.Ko₅ = Analyst.Ki₆ + Biologist.Ki₄ + Representative.Ki₁ – Explicit knowledge about data model
D7 Conceptual data model
1) Analyst.Ki₇ – Explicit knowledge about data model
1) Analyst.Ki₈ – Knowledge about ARIS tool
Combination
6. As-is conceptual data modeling with modeling tool
Roles: Analyst
1) Analyst.Ko₉ = Analyst.Ki₇ + Analyst.Ki₈ – Explicit knowledge about data model
D7 Conceptual data model

Fig. 9. Business process model developed according to proposed template

Table 4. Comparison of business process representation in BPMN and using of the proposed template

Criteria	*1*	*2*	*Description*
1 –BPMN 2 – Proposed template			
Input/output [data] Input/output [information] Input/output [knowledge]	- - -	+ + +	Unlike in other modeling languages, in the proposed model three flows are represented: information, knowledge, and data flow. The proposed template strictly distinguishes between tacit and explicit data, information, and knowledge. Artificial objects are the holders of explicit or materialized data, information, and knowledge; tacit data, information, and knowledge belonging to the certain person and are modeled inside the activity
Resource [knowledge]	-	+	In the proposed representation of a concept 'knowledge as a resource', which might be required for the role, is introduced. Knowledge as a resource unlike other types of resources (materials, technology) does not have depreciation, however training curve needs to be taken into account (time is needed to collect all required knowledge and skills).
Resource [human]	+/-	+	It is possible to add process performer to each activity. Knowledge is related to role (owner of knowledge) thus it is possible to derive specific knowledge associated to each person, as well as to trace how certain person obtains his/her knowledge during the process execution. The utilization of knowledge dimensions helps to plan the training and changes in required competences and resources already during business process modeling phase
Resource [artificial]	-	+	Resources, which are materialized and could be saved in knowledge repository as several documents, instructions, or books, are separated from the resources that are not materialized, but can facilitate process completion.
Resource [data store]	-	-	Proposed template does not provide separate modeling constructions for data store modeling
Knowledge intensive process type	-	+	Each activity is defined as either knowledge intensive or not knowledge intensive. In the template the knowledge conversion type is the attribute of an activity. It could be useful for improving knowledge sharing among process performers.
Process management	+	+	Has no specific benefits to compare to BPMN
Controls	+	+	Has no specific benefits to compare to BPMN
Control flows Decision points Events	+ + +	- - -	IDEF0 notation is not the most suitable for representing logic of the process, therefore, it should be combined with other notations that give more means for control and decision points modeling

- It gives a possibility to separate information and data in the business process modeling
- It gives an opportunity to identify the owner of data, information, and knowledge
- It gives a possibility to identify, plan, and manage knowledge of the role required for participating in a particular activity and linking this knowledge to the organizational competence model, if such is maintained
- It gives a possibility to evaluate the amount of lost organizational knowledge if a person – owner of knowledge – leaves the organization, i.e., to identify which tacit knowledge in which cases should be transformed into explicit knowledge, such as documents, rules, systems, etc.
- It gives an opportunity to improve understanding about the knowledge usefulness, validity, and relevance for particular activities in a process
- It gives an opportunity to enable competence requirements management and proactive training based on a business process analysis

The main disadvantages of the proposed approach are (1) complexity of graphical representation of the activity template and (2) drawbacks of procedural representation inherited from IDEF0. Therefore, further research will aim to design the appropriate means for reduction of complexity of representation and enriching model with possibilities to model events, decision points, and control flows or, alternatively, to link the template to the conventional business process models.

References

1. Baimin, B.S., Zijun, H., Xiohua, G.: Knowledge process reengineering and implementation of enterprise knowledge management. In: International Conference on Information Management, Innovation Management and Industrial Engineering, pp. 23–26 (2010)
2. Wang, S.-B., Wang, Ch , Yang, J. Research on the reengineering of government business processes based on the environment of e-government. In: International Conference on E-Business and E-Government, pp. 4503–4506 (2010)
3. Li, B.-Z., Liu, Y.: Organizational change pattern based on business process reengineering. In: International Conference on E-Business and E-Government, pp. 1193–1197 (2010)
4. Chalaris, I.E., Vlachopoulos, S.: Business Process Reengineering as a Modernizing Tool for the Public Administration- From Theory to Reality. In: Fourth Balkan Conference in Informatics, pp. 64–69 (2009)
5. Weicher, M., Chu, W.W., Lin, W.C., Le, V., Yu, D.: Business Process Reengineering: Analysis and Recommendations, `http://www.netlib.com/bpr1.htm#reenghr`
6. Businska, L., Supulniece, I., Kirikova, M.: On data, information, and knowledge representation in business process models. In: The 20th International Conference on Information Systems Development (ISD 2011), Edinburgh, Scotland. Springer (2011)
7. Supulniece, I., Businska, L., Kirikova, M.: Towards Extending BPMN with the Knowledge Dimension. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) BPMDS 2010 and EMMSAD 2010. LNBIP, vol. 50, pp. 69–81. Springer, Heidelberg (2010)

8. Янковский, С.Я.: Концепции общей теории информации. НиТ. Текущие публикации (2001), `http://n-t.ru/tp/ng/oti03.html`
9. Maier, R.: Knowledge management systems. Information and communication Technologies for knowledge management, 3rd edn. Springer, Heidelberg (2007)
10. Tiwana, A.: Knowledge Management Toolkit, The: Practical Techniques for Building a Knowledge Management System. Pearson Education (1999)
11. Beyon-Davies, P.B.: Significant threads: The nature of data. International Journal of Information Management 29, 170–188 (2009)
12. Francois, C. (ed.): International Encyclopedia of Systems and Cybernetics, 2nd edn. K.G. Saur, Munhen (2004)
13. Corning, P.A.: Control Information Theory: 'The missing link' in the science of cybernetics, Systems research and behavioral science. Syst. Res. 24, 297–311 (2007)
14. Beckman, T.: A methodology for knowledge management. In: IASTED International Conference on Artificial, Intelligence and Soft Computing (ASC 1997), Banff, Canada, pp. 29–32 (1997)
15. Gronau, N., Korf, R., Müller, C.: KMDL-Capturing, Analyzing and Improving Knowledge-Intensive Business Processes. Journal of Computer Science 4, 452–472 (2005)
16. Business Process Modeling Notation (BPMN), `http://www.omg.org/spec/BPMN/2.0/PDF`
17. GRADE Business Modeling, Language Reference, Infologistik GmbH (1998)
18. ARIS Expert Paper, Business Process Design as the Basis for Compliance Management, Enterprise Architecture and Business Rules (2007)
19. Arbeitsbericht (umfangreiche Beschreibung) - KMDL® v2.2, `http://www.kmdl.de`
20. IDEF0, `http://www.idef.com/IDEF0.html`
21. UML, `http://www.visual-paradigm.com/VPGallery/diagrams/Activity.html`

When Process Mining Meets Bioinformatics

R.P. Jagadeesh Chandra Bose[1,2] and Wil M.P. van der Aalst[1]

[1] Department of Mathematics and Computer Science, University of Technology, Eindhoven, The Netherlands
{j.c.b.rantham.prabhakara,w.m.p.v.d.aalst}@tue.nl
[2] Philips Healthcare, Veenpluis 5–6, Best, The Netherlands

Abstract. Process mining techniques can be used to extract non-trivial process-related knowledge and thus generate interesting insights from event logs. Similarly, bioinformatics aims at increasing the understanding of biological processes through the analysis of information associated with biological molecules. Techniques developed in both disciplines can benefit from one another, e.g., sequence analysis is a fundamental aspect in both process mining and bioinformatics. In this paper, we draw a parallel between bioinformatics and process mining. In particular, we present some initial success stories that demonstrate that the emerging process mining discipline can benefit from techniques developed for bioinformatics.

Keywords: sequence, trace, execution patterns, diagnostics, conformance, alignment, configuration.

1 Introduction

Bioinformatics aims at increasing the understanding of biological processes and entails the application of computational techniques to understand and organize the information associated with biological macromolecules [1]. Sequence analysis or sequence informatics is a core aspect of bioinformatics that is concerned with the analysis of DNA/protein sequences[1] and has been an active area of research for over four decades.

Process mining is a relatively young research discipline aimed at discovering, monitoring and improving real processes by extracting knowledge from event logs readily available in today's information systems [2]. Business processes leave trails in a variety of data sources (e.g., audit trails, databases, and transaction logs). Hence, every process instance can be described by a trace, i.e., a sequence of events. Process mining techniques are able to extract knowledge from such traces and provide a welcome extension to the repertoire of business process analysis techniques. The topics in process mining can be broadly classified into

[1] DNA stores information in the form of the base nucleotide sequence, which is a string of four letters (A, T, G and C) while protein sequences are sequences defined over twenty amino acids and are the fundamental determinants of biological structure and function.

three categories (i) *discovery*, (ii) *conformance*, and (iii) *enhancement* [2]. Process discovery deals with the discovery of models from event logs. For example, there are dozens of techniques that automatically construct process models (e.g., Petri nets or BPMN models) from event logs [2]. Discovery is not restricted to control-flow; one may also discover organizational models, etc. Conformance deals with comparing an apriori model with the observed behavior as recorded in the log and aims at detecting inconsistencies/deviations between a process model and its corresponding execution log. In other words, it checks for any violation between *what was expected to happen* and *what actually happened.* Enhancement deals with extending or improving an existing model based on information about the process execution in an event log. For example, annotating a process model with performance data to show bottlenecks, throughput times etc.

Despite several success stories there are still significant challenges that need to be addressed in applying process mining techniques on real-life event logs. Some of these challenges include:

– *Dealing with less structured processes:* most processes mined from real-life logs tend to be less structured than what stakeholders expect. The discovered process models are often spaghetti-like and are hard to comprehend. Many factors lead to such a behavior e.g., heterogeneity of cases, fine granular events, etc. Process models can be seen as "maps" describing the operational processes of organizations. There is a need for techniques that enable the discovery of *navigable* process maps with seamless zoom-in/zoom-out facility (hierarchical process models with different perspectives).
– *Dealing with fine granular event logs:* some event logs (especially those that emanate from large scale processes, high-tech systems such as medical systems, copiers and scanners, etc) contain events at a very low abstraction level. Stakeholders would like to view processes at a more coarse-grained level. There is a need for (semi-)automated means of aggregating low-level events into high-level events. Voluminous data is a natural side effect of such fine granular event logs. This imposes an additional requirement on the process mining techniques to be scalable as well.
– *Provisions for process diagnostics:* The lion's share of process mining research has been devoted to control-flow discovery. Process diagnostics, which encompasses process conformance checking, auditing, process performance analysis, anomaly detection, diagnosis, inspection of interesting patterns and the like, is gaining prominence in recent years [3,4,5,6,7]. There is a need for techniques that assist auditors and analysts in their diagnostic efforts [8].
– *Dealing with process changes:* contemporary process mining techniques assume the processes to be in steady state. However, in reality, processes may change to adapt to changing circumstances, e.g., new legislation, extreme variations in supply and demand, seasonal effects, etc. *Concept drift* refers to the situation in which the process is changing while being analyzed [9]. There is a need for techniques that deal with such "second order dynamics". Analyzing such changes is of utmost importance to get an accurate insight on process executions at any instant of time.

It is important to note that, to a large extent, *sequence analysis is a fundamental aspect in almost all facets of process mining and bioinformatics*. In spite of all the peculiarities specific to business processes and process mining, the relatively young field of process mining should, in our view, take account of the conceptual foundations, practical experiences, and analysis tools developed by sequence informatics researchers over the last couple of decades. In this paper, we describe some of the analogies between the problems studied in both disciplines. We present some initial successes which demonstrate that process mining techniques can benefit from such a cross-fertilization.

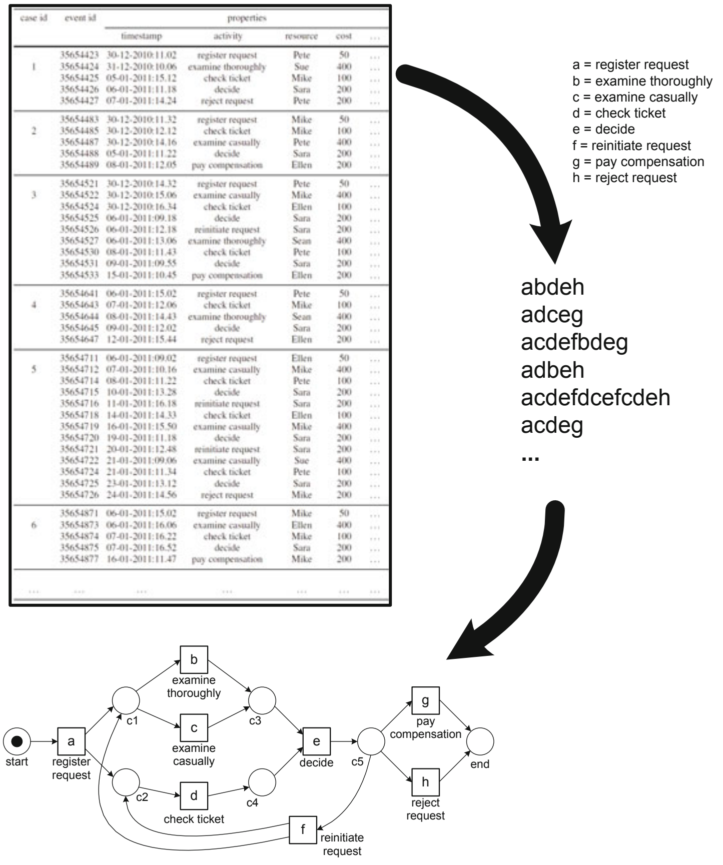
The remainder of this paper is organized as follows. Section 2 introduces some of the basic process mining concepts and illustrates some of the challenges already mentioned. The subsequent sections relate ideas and techniques from bioinformatics to process mining. Section 3 points out similarities in the structuring mechanisms used in both domains, e.g., the hierarchy of protein structures is compared to the hierarchical structuring of events in processes. Section 4 discusses commonalities between alignments in biology and traces in event logs. Section 5 relates phylogeny (the creation of tree structures showing inferred evolutionary relationships among various biological species) to process configuration. Section 6 concludes the paper.

2 Preliminaries: Process Mining

The goal of this paper is to show that process mining can benefit from ideas and techniques originating from bioinformatics. However, before doing so, we first introduce some of the basic process mining concepts and illustrate that there are indeed several problems to be tackled.

Process mining serves a bridge between data mining and business process modeling. The goal is to extract process-related knowledge from event data recorded by a variety of systems (ranging from sensor networks to enterprise information systems). Starting point for process mining is an event log. We assume that events can be related to process instances (often called cases) and are described by some activity name. The events within a process instance are ordered. Therefore, a process instance is often represented as a trace over a set of activities. In real-life event logs, events have timestamps, associated resources (e.g. the person executing the activity), transactional information (e.g., start, complete, or suspend), data attributes (e.g., amount or type of customer). However, for clarity, we abstract from such additional information. Therefore, we can use the following basic notations:

– Σ denotes the set of *activities*. Σ^+ is the set of all non-empty finite sequences of activities from Σ.
– A *process instance* (i.e. case) is described as a *trace* over Σ, i.e., a finite sequence of activities. Examples of traces are `abcd` and `abbbad`.
– Let $T = T(1)T(2)T(3)\dots T(n) \in \Sigma^+$ be a trace over Σ. $T(k)$ represents the k^{th} activity in the trace. $|T| = n$ denotes the *length* of the trace T.



case id	event id	properties				
		timestamp	activity	resource	cost	...
1	35654423	30-12-2010:11.02	register request	Pete	50	...
	35654424	31-12-2010:10.06	examine thoroughly	Sue	400	...
	35654425	05-01-2011:15.12	check ticket	Mike	100	...
	35654426	06-01-2011:11.18	decide	Sara	200	...
	35654427	07-01-2011:14.24	reject request	Pete	200	...
2	35654483	30-12-2010:11.32	register request	Mike	50	...
	35654485	30-12-2010:12.12	check ticket	Mike	100	...
	35654487	30-12-2010:14.16	examine casually	Pete	400	...
	35654488	05-01-2011:11.22	decide	Sara	200	...
	35654489	08-01-2011:12.05	pay compensation	Ellen	200	...
3	35654521	30-12-2010:14.32	register request	Pete	50	...
	35654522	30-12-2010:15.06	examine casually	Mike	400	...
	35654524	30-12-2010:16.34	check ticket	Ellen	100	...
	35654525	06-01-2011:09.18	decide	Sara	200	...
	35654526	06-01-2011:12.18	reinitiate request	Sara	200	...
	35654527	06-01-2011:13.06	examine thoroughly	Sean	400	...
	35654530	08-01-2011:11.43	check ticket	Pete	100	...
	35654531	09-01-2011:09.55	decide	Sara	200	...
	35654533	15-01-2011:10.45	pay compensation	Ellen	200	...
4	35654641	06-01-2011:15.02	register request	Pete	50	...
	35654643	07-01-2011:12.06	check ticket	Mike	100	...
	35654644	08-01-2011:14.43	examine thoroughly	Sean	400	...
	35654645	09-01-2011:12.02	decide	Sara	200	...
	35654647	12-01-2011:15.44	reject request	Ellen	200	...
5	35654711	06-01-2011:09.02	register request	Ellen	50	...
	35654712	07-01-2011:10.16	examine casually	Mike	400	...
	35654714	08-01-2011:11.22	check ticket	Pete	100	...
	35654715	10-01-2011:13.28	decide	Sara	200	...
	35654716	11-01-2011:16.18	reinitiate request	Sara	200	...
	35654718	14-01-2011:14.33	check ticket	Ellen	100	...
	35654719	16-01-2011:15.50	examine casually	Mike	400	...
	35654720	19-01-2011:11.18	decide	Sara	200	...
	35654721	20-01-2011:12.48	reinitiate request	Sara	200	...
	35654722	21-01-2011:09.06	examine casually	Sue	400	...
	35654724	21-01-2011:11.34	check ticket	Pete	100	...
	35654725	23-01-2011:13.12	decide	Sara	200	...
	35654726	24-01-2011:14.56	reject request	Mike	200	...
6	35654871	06-01-2011:15.02	register request	Mike	50	...
	35654873	06-01-2011:16.06	examine casually	Ellen	400	...
	35654874	07-01-2011:16.22	check ticket	Mike	100	...
	35654875	07-01-2011:16.52	decide	Sara	200	...
	35654877	16-01-2011:11.47	pay compensation	Mike	200	...
...	...	...	...	...	...	...

Fig. 1. Process discovery aims to learn a process model (in this case a Petri net) from traces of activities

– An *event log*, $\mathcal{L}$, corresponds to a multi-set (or bag) of traces from Σ^+. For example, $\mathcal{L} = [\texttt{abcd}, \texttt{abcd}, \texttt{abbbad}]$ is a log consisting of three cases. Two cases follow trace `abcd` and one case follows trace `abbbad`.

As mentioned in Section 1, event logs can be used to conduct three types of process mining: (i) discovery, (ii) conformance, and (iii) enhancement [2]. Process discovery—discovering a process model from example behavior recorded in an event log—is one of the most challenging tasks in process mining. Today there are dozens of process discovery techniques generating process models using different notations (Petri nets, EPCs, BPMN, heuristic nets, etc.). Fig. 1 illustrates the basic idea of process discovery. An event log containing detailed information about events is transformed into a multiset of traces $\mathcal{L} = [\texttt{abdeh}, \texttt{adceg}, \texttt{acdefbdeg}, \texttt{adbeh}, \texttt{acdefdcefcdeh}, \texttt{acdeg}, \ldots]$. Process discovery techniques are able to discover process models such as the Petri net shown in Fig. 1.

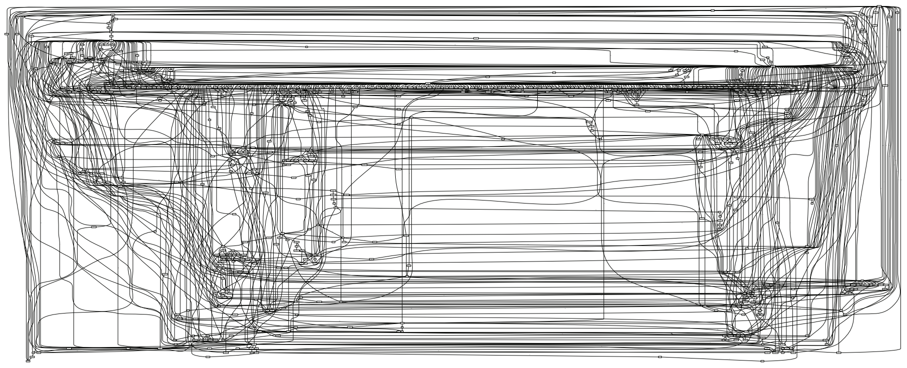


Fig. 2. Spaghetti process describing the diagnosis and treatment of 2765 patients in a Dutch hospital. The process model was constructed based on an event log containing 114,592 events. There are 619 different activities (taking event types into account) executed by 266 different individuals (doctors, nurses, etc.).

Event logs may be *incomplete* and contain *noise*. Noise refers to rare and infrequent behavior not representative for the typical behavior of the process. Incompleteness refers to the problem that one typically sees only a fraction of all possible behaviors. Traces that are not seen in the log are not necessarily impossible; we only see positive examples and no negative examples. Process mining algorithms need to be able to deal with noise and incompleteness. Generally, we use four main quality dimensions for judging the quality of the discovered process model: *fitness*, *simplicity*, *precision*, and *generalization* [2]. A model with good fitness allows for the behavior seen in the event log. The simplest model that can explain the behavior seen in the log, is the best model (Occam's Razor). A model that is not precise is "underfitting". Underfitting is the problem that the model over-generalizes the example behavior in the log, i.e., the model allows for behaviors very different from what was seen in the log. A model that does not generalize is "overfitting". Overfitting is the problem that a very specific model is generated whereas it is obvious that the log only holds example behavior, i.e., the model explains the particular sample log, but a next sample log of the same process may produce a completely different process model.

The challenges related to process mining are best explained using an example. Fig. 2 shows an example of a typical Spaghetti process discovered using conventional process mining techniques [2]. The complexity of the diagram illustrates the problems and challenges mentioned in Section 1. In the remainder of the paper, we show how ideas and techniques originating from bioinformatics can help to address these.

3 From Sequence to Structure

A DNA *sequence motif* is defined as a nucleic acid *sequence pattern* that has some biological significance (both structural and functional) [10]. These motifs

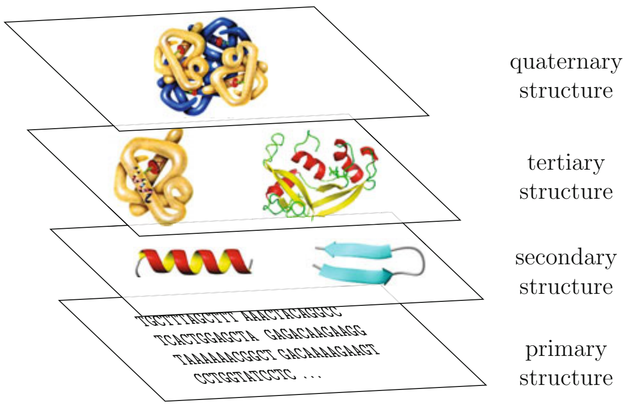


Fig. 3. Hierarchy of protein structures

are usually found to recur in different genes or within a single gene. For example, *tandem repeats* (tandemly repeating DNA) are associated with various regulatory mechanisms such as protein binding [11]. More often than not, sequence motifs are also associated with *structural motifs* found in proteins thus establishing a strong correspondence between sequence and structure. Protein structures manifest as a hierarchy of four levels: primary, secondary, tertiary, and quaternary. Primary structure is the basic level and corresponds to the linear sequence of amino acids. Secondary structures result from the regular folding of regions within the amino acid sequence into particular structural patterns e.g., α-helix, β-sheets, β-turns, loops, etc. Tertiary and quaternary structures result from the folding of primary structure and secondary structural elements in 3 dimensions. Fig. 3 depicts the hierarchy of protein structures.

Likewise, common subsequences of activities in an event log that are found to recur within a process instance or across process instances have some domain (functional) significance. In [12], we adopted the sequence patterns (e.g., tandem repeats, maximal repeats etc.) proposed in the bioinformatics literature, correlated them to commonly used process model constructs (e.g., tandem repeats and tandem arrays correspond to simple loop constructs), and proposed a means to form abstractions over these patterns. The abstractions thus uncovered have a strong domain significance from a functionality point of view. Using these abstractions as a basis, we proposed a *two-phase approach to process discovery* [13]. The first phase comprises of pre-processing the event log with abstractions at a desired level of granularity and the second phase deals with discovering the *process maps* with seamless zoom-in/out facility. Fig. 4 summarizes the overall approach. Note the similarity with Fig. 3.

Fig. 5 highlights the difference between the traditional approach to process discovery and the two-phase approach. Note that the process model (map) discovered using the two-phase approach is simpler. Our approach supports the abstraction of activities based on their context and type, and provides a seamless zoom-in and zoom-out functionality. Fig. 5 illustrates that a cross-fertilization

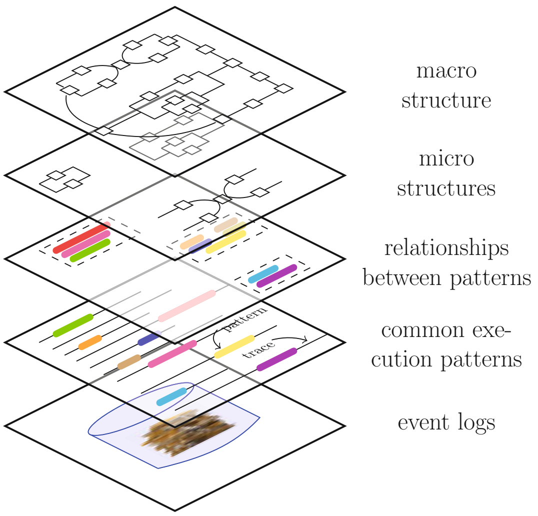
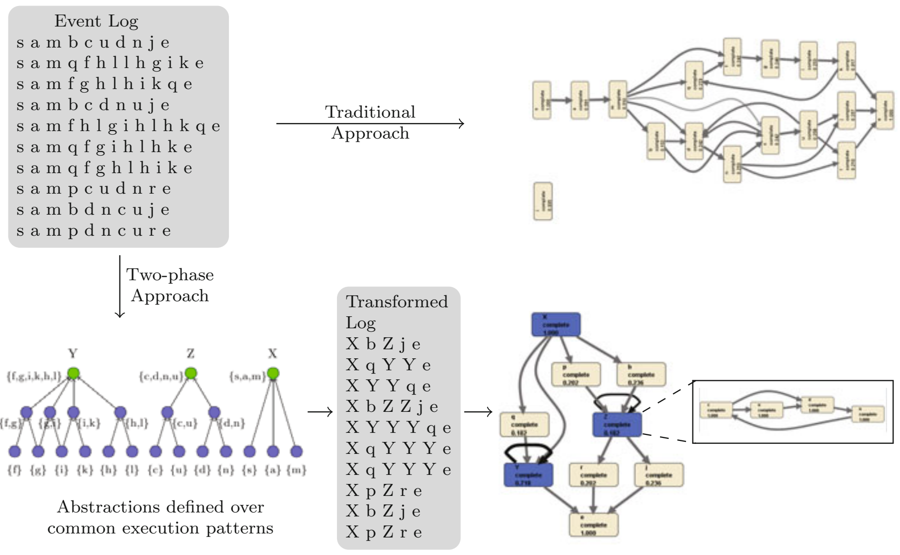


Fig. 4. Repeating subsequences of activities define the common execution patterns and carry some domain (functional) significance. Related patterns and activities pertaining to these patterns define abstractions that correspond to micro-structures (or sub-processes). The top-level process model can be viewed as a macro-structure that subsumes the micro-structures.


Event Log
s a m b c u d n j e
s a m q f h l l h g i k e
s a m f g h l h i k q e
s a m b c d n u j e
s a m f h l g i h l h k q e
s a m q f g i h l h k e
s a m q f g h l h i k e
s a m p c u d n r e
s a m b d n c u j e
s a m p d n c u r e

Transformed Log
X b Z j e
X q Y Y e
X Y Y q e
X b Z Z j e
X Y Y Y q e
X q Y Y Y e
X q Y Y Y e
X p Z r e
X b Z j e
X p Z r e

Traditional Approach
Two-phase Approach
Y
Z
X
{f,g,i,k,h,l}
{c,d,n,u}
{s,a,m}
{f,g}
{g,i}
{i,k}
{h,l}
{c,u}
{d,n}
{f} {g} {i} {k} {h} {l} {c} {u} {d} {n} {s} {a} {m}
Abstractions defined over common execution patterns

Fig. 5. Traditional approach vs. our two-phase approach to process discovery

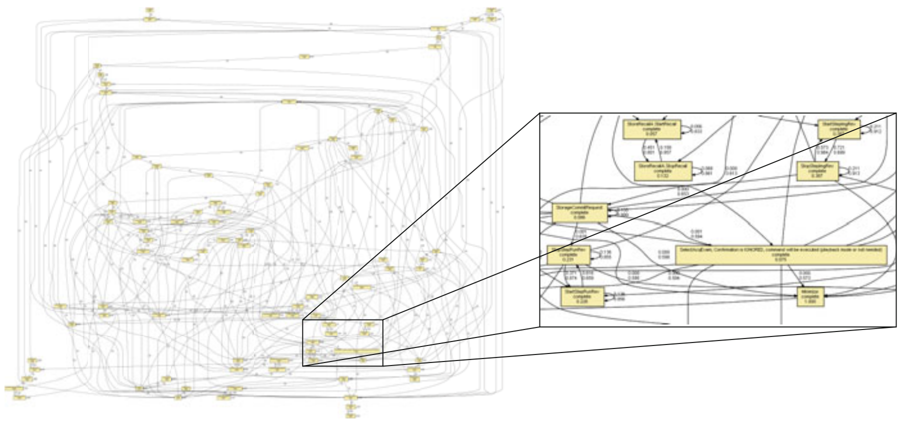


Fig. 6. Process model discovered using the classical Fuzzy miner on the event log capturing the activities of field service engineers during one of the part replacements in X-ray machines

between bioinformatics and process mining enables the discovery of hierarchical process models. This provides a new perspective when dealing with fine granular event logs and less structured processes.

We next present the results of applying this approach on a real-life case study. The case study was performed in close collaboration with Philips Healthcare and pertains to the analysis of event logs generated by the X-ray machines of Philips used all over the globe. More specifically, we considered an event log capturing the activities performed by field service engineers during one of the part replacements on X-ray machines. The event log contains 113 cases and 76,754 events referring to 92 activities. Fig. 6 depicts the process model uncovered on this log using the classical Fuzzy miner. We can see an incomprehensible spaghetti-like model. We uncover common execution patterns, define abstractions, and use our two-phase approach to alleviate this problem. The transformed log (with abstractions) contains 113 cases and 10,387 events referring to 20 activities. Fig. 7 depicts the top-level process model discovered using the Fuzzy map miner on this transformed log. We can see that the model discovered using our two-phase approach is simpler and more comprehensible. We can explore the abstract nodes (blue (dark) colored) to see the sub-processes captured underneath them. For example, Fig. 8(a) depicts the sub-process for the abstract activity Examination Administration and Viewing while Fig. 8(b) depicts the sub-process for the abstract activity Dynamic Viewing.

We have used this approach on several other real-life event logs as well (e.g., see [14]) and in all of these case studies, our approach is able to mine meaningful (from a domain point of view) and comprehensive models. However, given the space limitations and the scope of the paper, we cannot elaborate more on these applications.

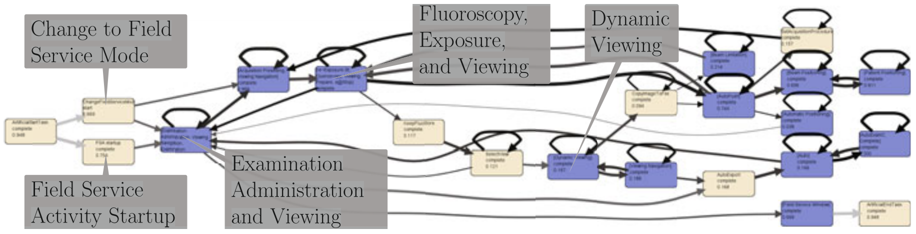
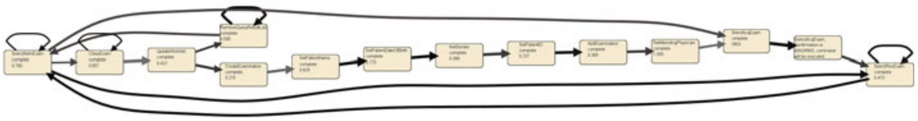
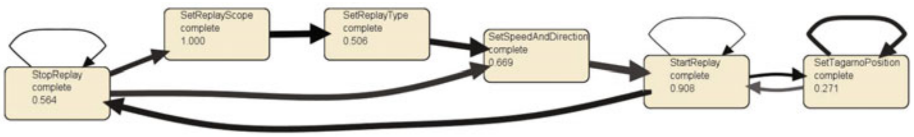

Change to Field Service Mode
Fluoroscopy, Exposure, and Viewing
Dynamic Viewing
Field Service Activity Startup
Examination Administration and Viewing

Fig. 7. The process model discovered using our two-phase approach. Blue (dark) colored nodes signify abstract activities that can be zoomed in to view the sub-processes underneath it.



(a) Examination Administration and Viewing


SetReplayScope complete 1.000
SetReplayType complete 0.506
SetSpeedAndDirection complete 0.669
StopReplay complete 0.564
StartReplay complete 0.908
SetTagarnoPosition complete 0.271

(b) Dynamic Viewing

Fig. 8. The sub-processes corresponding to the abstract activities Examination Administration and Viewing and Dynamic Viewing

4 Sequence Alignment and Process Diagnostics

Multiple sequence alignment has been a subject of extensive research in computational biology for over three decades. Sequence alignment is an essential tool in bioinformatics that assists in unraveling the secondary and tertiary structures of proteins and molecules, their evolution and functions, and in inferring the taxonomic, phylogenetic or cladistic relationships between organisms, diagnoses of genetic diseases, etc. [15,16].

In [17], we have adapted sequence alignment to traces in an event log and showed that it carries significant promise in process diagnostics. The goal of *trace alignment* is to align traces in such a way that event logs can be easily explored. Given a multi-set of traces $\mathbb{T} = [T_1, T_2, \ldots, T_n]$, trace alignment can be defined as a mapping of $\mathbb{T}$ to another multi-set of traces $\overline{\mathbb{T}} = [\overline{T_1}, \overline{T_2}, \ldots, \overline{T_n}]$ where $\overline{T_i} \in (\Sigma \cup \{-\})^+$ for $1 \leq i \leq n$. In addition, the following three properties need to be satisfied with respect to $\mathbb{T}$ and $\overline{\mathbb{T}}$:

– each trace in $\overline{\mathbb{T}}$ is of the same length i.e., there exists an $m \in \mathbb{N}$ such that $|\overline{T_1}| = |\overline{T_2}| = \cdots = |\overline{T_n}| = m$
– $\overline{T_i}$ is equal to T_i after removing all gap symbols '$-$' and

– there is no $k \in \{1, \ldots, m\}$ such that $\forall_{1 \leq i \leq n}\ \overline{T_i}(k) = -$.

Trace alignment can be used to explore the process in the early stages of analysis and to answer specific questions in later stages of analysis. Fig. 9 depicts the results of trace alignment for a real-life log from a rental agency[2]. Every row corresponds to a process instance and time increases from left to right. The horizontal position is based on *logical time* rather than real timestamps. If two rows have the same activity name in the same column, then the corresponding two events are very similar and are therefore aligned. Note that the same activity can appear in multiple columns. By reading a row from left to right, we can see the sequence of activities (i.e., the trace) that was executed for a process instance. Process instances having the same trace can be grouped into one row to simplify the diagram. The challenge is to find an alignment that is as simple and informative as possible. For example, the number of columns and gaps should be minimized while having as much consensus as possible per column.

Trace alignment can assist in answering a variety of diagnostic questions. For example, one can get answers to questions such as:

– *What is the most common (likely) process behavior that is executed?*
  The consensus sequence of an alignment, which captures the major activity in each column, represents the most common process behavior that is executed and can be considered as the back-bone sequence for the process.
– *Are there any common patterns of execution in my traces?*
  Common execution patterns are captured in the form of well conserved regions (columns) in the alignment. For example, the activity sequence `b0e0a5` (at columns 5−7) corresponding to the activities, `planning of first inspection`, `preparation of lease termination form`, and `is first inspection performed?` respectively, is common across all the traces.
– *Where do my process instances deviate and what do they have in common?*
  Deviations, exceptional behavior and rare event executions are captured in regions that are sparsely filled i.e., regions with lot of gap symbols ($-$) or in regions that are well conserved with a few rare gaps.
  For example, it could be seen that only one of the traces (sixth trace in the alignment) has the activity subsequence `a9e9a5` in columns $8 - 10$. Activity `a5` in column 7 corresponds to the check, `is first inspection performed?` and the activity subsequence `a9e9a5` corresponds to the scenario where the result of the check was negative due to the fact that the tenant was not at home. `a9` corresponds to the activity of sending a letter to the tenant and `e9` corresponds to the activity of rescheduling the first inspection.
– *What are the contexts in which an activity or a set of activities is executed in my event log?*
  Trace alignment provides a complete perspective of activity executions in a log including that of long range dependencies (any dependencies between activities are reflected as common execution patterns in the traces where

[2] Since the whole alignment is not legible, we highlight the interesting patterns/activities at the top and the bottom of the figure.

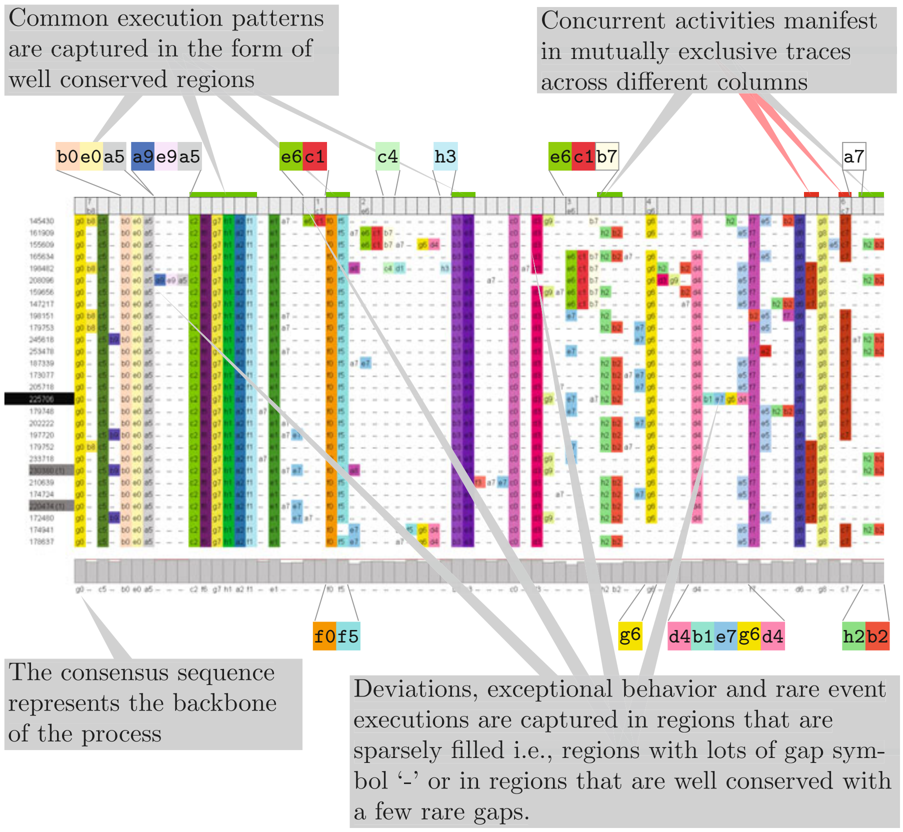

Common execution patterns are captured in the form of well conserved regions
Concurrent activities manifest in mutually exclusive traces across different columns
b0 e0 a5
a9 e9 a5
e6 c1
c4
h3
e6 c1 b7
a7
f0 f5
g6
d4 b1 e7 g6 d4
h2 b2
The consensus sequence represents the backbone of the process
Deviations, exceptional behavior and rare event executions are captured in regions that are sparsely filled i.e., regions with lots of gap symbol '-' or in regions that are well conserved with a few rare gaps.

Fig. 9. An example of trace alignment for a real-life log from a rental agency. Each row refers to a process instance. Columns describe positions in traces. Consider now the cell in row y and column x. If the cell contains an activity name `a`, then `a` occurred for case y at position x. If the cell contains no activity name (i.e., a gap "–"), then nothing happened for y at position x.

they manifest). Furthermore, with rich interactive visualization (such as the options of filtering columns containing an activity), trace alignment enables a flexible inspection of the log.

– *What are the process instances that share/capture a desired behavior either exactly or approximately?*
  One can formulate the desired behavior as an activity sequence and apply trace alignment of this sequence with the traces in the log. Traces/process instances that share the desired behavior have a lot of their activities aligned with that of the activities in the desired behavior sequence.
– *Are there particular patterns (e.g., milestones, concurrent activities etc.) in my process?*
  Concurrent activities manifest in mutually exclusive traces across different

columns in an alignment. For example, the activities `h2b2` corresponding to the `drafting of final note` (`h2`) and `archiving of lease termination` (`b2`) is concurrent in this process.

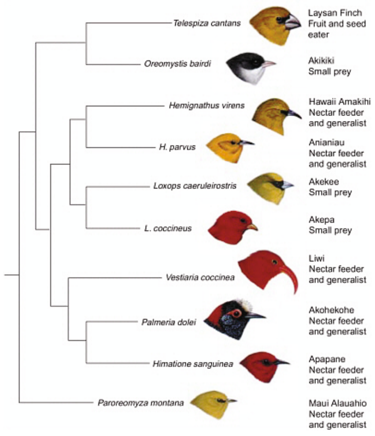
The application of sequence alignment in bioinformatics to process mining has created an altogether new dimension to conformance checking; *deviations and violations are uncovered by analyzing just the raw event traces* (thereby avoiding the need for process models).

Finding good quality alignments is notoriously complex. The initial results of trace alignment are definitely encouraging. Nonetheless, there are various new challenges when adopting biological sequence alignment to trace alignment in the context of business processes [18]. For example, biological sequences tend to be homogenous whereas traces in semi-structured processes (e.g., care processes in hospitals) tend to be heterogeneous. Other differences are the fact that traces in an event log can be of very different lengths (e.g., due to loops) and may be the result of concurrency. These characteristics provide new challenges for sequence alignment.

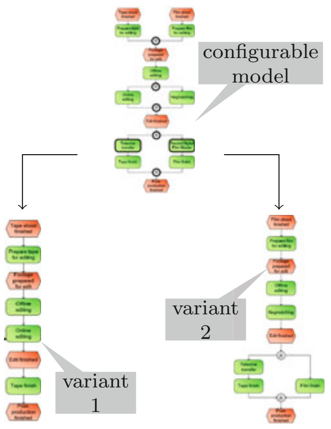
5 Phylogeny and Process Configuration

Phylogenetics refers to the study of evolutionary relationships, and was one of the first applications in bioinformatics. A phylogeny is a tree representation of the evolutionary history of a set (family) of organisms, gene/protein sequences etc. The basic premise in phylogenetics is that genes have evolved by duplication and divergence from common ancestors [19]. The genes can therefore exist in a nested hierarchy of relatedness. Fig. 10(a) depicts the phylogeny of some of the species of Hawaiian honeycreeper [20]. These variant species descended from a single species over the last ten million years.

Phylogeny is related to structuring variability within and between processes. In the past couple of years, *process configuration* has gained prominence in the BPM community [21]. Process configuration is primarily concerned with managing families of business processes that are similar to one another in many ways yet differing in some other ways. For example, processes within different municipalities are very similar in many aspects and differ in some other aspects. Such discrepancies can arise due to characteristics peculiar to each municipality (e.g., differences in size, demographics, problems, and policies) that need to be maintained. Furthermore, operational processes need to change to adapt to changing circumstances, e.g., new legislation, extreme variations in supply and demand, seasonal effects, etc. A configurable process model describes a family of similar process models in a given domain [21], and can be thought of as the genesis (root) of the family. All variants in the family can be derived from the configurable model through a series of change patterns [22] and configuration patterns [23]. Fig. 10(b) depicts an example of a configurable model (parent) and two variants (children) derived from it. One of the core research problems in *process configuration* is to automatically derive configurable process models from specific models and event logs.


Telespiza cantans
Laysan Finch Fruit and seed eater
Oreomystis bairdi
Akikiki Small prey
Hemignathus virens
Hawaii Amakihi Nectar feeder and generalist
H. parvus
Anianiau Nectar feeder and generalist
Loxops caeruleirostris
Akekee Small prey
L. coccineus
Akepa Small prey
Vestiaria coccinea
Liwi Nectar feeder and generalist
Palmeria dolei
Akohekohe Nectar feeder and generalist
Himatione sanguinea
Apapane Nectar feeder and generalist
Paroreomyza montana
Maui Alauahio Nectar feeder and generalist

(a) Phylogeny of the Hawaiian honeycreeper.



(b) Process configuration

Fig. 10. Similarity between phylogeny and process configuration

One can find stark similarity between phylogenetics and process configuration. Techniques have been proposed in the bioinformatics literature to discover phylogenies both from (protein) structure as well as from sequences. This can be compared to deriving configurable process models from specific models and from event logs respectively. The adaptability of phylogeny construction techniques to process configuration needs to be explored.

Techniques from bioinformatics have also been adopted to *trace clustering* in process mining [24,25]. Trace clustering was shown to be effective in dealing with the heterogeneity in event logs [24,25]. Process mining results can be improved by segregating heterogeneous cases into more homogenous clusters and analyzing each cluster separately. Sequence clustering techniques have been applied to deal with unlabeled event logs[3] in process mining [26]. Experiences from bioinformatics can also contribute to tooling and infrastructure efforts in process mining. For example, visualization is one of the challenging problems in process mining tooling. A lot of current visualization means in process mining tools such as ProM[4] become unmanageable when dealing with large event logs thereby compromising the comprehensibility. Process mining is typically an iterative activity driven by questions from stakeholders and surprising analysis results. Techniques for visualization in process mining should focus on support-

[3] In an unlabeled event log, the case to which an event belongs to is unknown.

[4] ProM is an extensible framework that provides a comprehensive set of tools/plugins for the discovery and analysis of process models from event logs. See `http://www.processmining.org` for more information and to download ProM.

ing the strong iterative and interactive nature of event log analysis e.g., ranging from overview results to focused and directed insights, annotating mined results, enabling holistic views by juxtaposing several different analysis results simultaneously, etc. *Visualization* is used in many areas within bioinformatics (e.g., sequence matching, genome browsing, multiple sequence alignment, etc.), with varying success, and good tools already exist. There is significant potential to learn from the success stories in bioinformatics, e.g., event logs refer to multi-sets of traces, which are basically collections of sequences; sequence exploration and visualization techniques in bioinformatics can be assessed for their adoption to event logs.

Benchmarking and *data repositories* form another area where bioinformatics has matured over the years. To cater to the rapidly increasing accumulation of biological data, lots of efforts had been initiated in bioinformatics to create advanced databases with analysis capabilities devoted to particular categories e.g., Genbank (cataloguing DNA data), SWISS-PROT/TrEMBL (repository of protein sequences), etc. These repositories support features such as protein sequence/structural/functional comparison and classification benchmarks. Process mining being an emerging technology, such repositories and good benchmarks are still missing. Recently, several efforts had been initiated in the process modeling and process mining community to create repositories with advanced support for dealing with process model collections e.g., APROMORE [27], and repositories of event logs [28]. Process mining repositories and benchmarks should include:

- event logs and process mining tasks e.g., control-flow discovery, organizational model extraction, etc.
- event logs, process models and associated tasks e.g., process conformance, replay techniques, etc.
- process models with associated characteristics e.g., functional (such as loan application process), structural (such as the workflow patterns present), behavioral, etc.

Event log and process model comparison methods, search, and exploration are some of the essential features that these repositories need to support. Quality metrics (e.g., fitness, precision, generalization, computational complexity, etc.) of state-of-the-art techniques also need to be captured in these repositories. This enables the comparison of performance of a new algorithm/technique with existing methods. It is also desirable to elicit validation protocols to streamline the ways in which such quality metrics are measured.

Such an overlap between the goals combined with the promising initial results calls for a more rigorous attempt at understanding and exploiting the synergy between these two disciplines.

6 Conclusions

Bioinformatics and process mining share some common goals. In this paper, we presented the commonalities between the problems and techniques studied in bioinformatics and process mining. The commonalities can be exploited: process mining is a relative young discipline that can benefit from the plethora of techniques developed in bioinformatics. In this paper, we presented some initial results that show that such a cross-fertilization between both areas is indeed beneficial. However, as indicated, the examples in this paper are just initial steps and further exploration of the topic is needed.

Acknowledgments. The authors are grateful to Philips Healthcare for funding the research in process mining.

References

1. Luscombe, N., Greenbaum, D., Gerstein, M.: What is Bioinformatics? A Proposed Definition and Overview of the Field. Methods of Information in Medicine 40(4), 346–358 (2001)
2. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
3. Rozinat, A., van der Aalst, W.M.P.: Conformance Checking of Processes Based on Monitoring Real Behavior. Information Systems 33(1), 64–95 (2008)
4. van der Aalst, W.M.P., van Hee, K.M., van der Werf, J.M., Verdonk, M.: Auditing 2.0: Using Process Mining to Support Tomorrow's Auditor. Computer 43(3), 90–93 (2010)
5. van der Aalst, W.M.P., de Medeiros, A.K.A.: Process Mining and Security: Detecting Anamolous Process Executions and Checking Process Conformance. Electronic Notes in Theoretical Computer Science 121, 3–21 (2005)
6. Yang, W.S., Hwang, S.Y.: A Process Mining Framework for the Detection of Healthcare Fraud and Abuse. Expert Systems with Applications 31(1), 56–68 (2006)
7. Bezerra, F., Wainer, J., van der Aalst, W.M.P.: Anomaly Detection Using Process Mining. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) BPMDS 2009 and EMMSAD 2009. LNBIP, vol. 29, pp. 149–161. Springer, Heidelberg (2009)
8. van der Aalst, W.M.P.: Challenges in Business Process Mining. Technical Report BPM-10-01, Business Process Management (BPM) Center (2010)
9. Bose, R.P.J.C., van der Aalst, W.M.P., Žliobaitė, I., Pechenizkiy, M.: Handling Concept Drift in Process Mining. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 391–405. Springer, Heidelberg (2011)
10. Das, M.K., Dai, H.K.: A Survey of DNA Motif Finding Algorithms. BMC Bioinformatics 8(suppl. 7), S21 (2007)
11. Kolpakov, R., Bana, G., Kucherov, G.: mreps: Efficient and Flexible Detection of Tandem Repeats in DNA. Nucleic Acids Research 31(13), 3672–3678 (2003)
12. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Abstractions in Process Mining: A Taxonomy of Patterns. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 159–175. Springer, Heidelberg (2009)
13. Li, J., Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Mining Context-Dependent and Interactive Business Process Maps Using Execution Patterns. In: zur Muehlen, M., Su, J. (eds.) BPM 2010 Workshops. LNBIP, vol. 66, pp. 109–121. Springer, Heidelberg (2011)

14. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Analysis of Patient Treatment Procedures: The BPI Challenge Case Study. Technical Report BPM-11-18, BPMCenter.org (2011), http://bpmcenter.org/wp-content/uploads/reports/2011/BPM-11-18.pdf
15. Chan, S., Wong, A.K.C., Chiu, D.: A Survey of Multiple Sequence Comparison Methods. Bulletin of Mathematical Biology 54(4), 563–598 (1992)
16. Gotoh, O.: Multiple Sequence Alignment: Algorithms and Applications. Advanced Biophysics 36, 159–206 (1999)
17. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Trace Alignment in Process Mining: Opportunities for Process Diagnostics. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 227–242. Springer, Heidelberg (2010)
18. Notredame, C.: Recent Progress in Multiple Sequence Alignment: A Survey. Pharmacogenomics 3, 131–144 (2002)
19. Thornton, J.W., DeSalle, R.: Gene Family Evolution and Homology: Genomics Meets Phylogenetics. Annual Review of Genomics and Human Genetics 1(1), 41–73 (2000)
20. Olson, S.: Evolution in Hawaii: A Supplement to Teaching About Evolution and the Nature of Science. National Academic Press (2004)
21. van der Aalst, W.M.P., Lohmann, N., Rosa, M.L., Xu, J.: Correctness Ensuring Process Configuration: An Approach Based on Partner Synthesis. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 95–111. Springer, Heidelberg (2010)
22. Weber, B., Rinderle, S., Reichert, M.: Change Patterns and Change Support Features in Process-Aware Information Systems. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 574–588. Springer, Heidelberg (2007)
23. Dreiling, A., Rosemann, M., van der Aalst, W.M.P., Heuser, L., Schulz, K.: Model-Based Software Configuration: Patterns and Languages. European Journal of Information Systems 15(6), 583–600 (2006)
24. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Context Aware Trace Clustering: Towards Improving Process Mining Results. In: Proceedings of the SIAM International Conference on Data Mining (SDM), pp. 401–412 (2009)
25. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Trace Clustering Based on Conserved Patterns: Towards Achieving Better Process Models. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009. LNBIP, vol. 43, pp. 170–181. Springer, Heidelberg (2010)
26. Ferreira, D., Zacarias, M., Malheiros, M., Ferreira, P.: Approaching Process Mining with Sequence Clustering: Experiments and Findings. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 360–374. Springer, Heidelberg (2007)
27. Rosa, M.L., Reijers, H.A., van der Aalst, W.M.P., Dijkman, R.M., Mendling, J., Dumas, M., Garcia-Banuelos, L.: APROMORE: An Advanced Process Model Repository. Expert Systems with Applications 38(6), 7029–7040 (2011)
28. 3TU.DataCentrum: http://data.3tu.nl/repository/collection:event_logs

Stepwise Context Boundary Exploration Using Guide Words

Naoyasu Ubayashi and Yasutaka Kamei

Kyushu University, Japan
ubayashi@acm.org, kamei@ait.kyushu-u.ac.jp

Abstract. Most requirements elicitation methods do not explicitly provide a systematic way for deciding the boundary of the usage context that should be taken into account because it is essentially difficult to decide which context element should be included as the system requirements. If a developer explores the context boundary in an ad-hoc manner, the developer will be faced with the *frame problem* because there are unlimited context elements in the real world where the target system exists. There are many application domains that should take into account the *frame problem*: security, safety, network threats, and user interactions. To deal with this problem, this paper proposes a new type of requirements analysis method for exploring the context boundary using guide words, a set of hint words for finding a context element affecting the system behavior. The target of our method is embedded systems that can be abstracted as a sensor-and-actuator machine exchanging the physical value between a system and its context. In our method, only the *value-context elements*, a kind of *value objects*, are extracted as the associated context elements. By applying the *guide words*, we can explore only a sequence of context elements affecting the data value and avoid falling into the *frame problem* at the requirements analysis phase.

Keywords: Context analysis, Frame problem, Embedded systems.

1 Introduction

Many embedded systems not only affect their context through actuators but also are affected by their context through sensors. The term *context* refers to the real world such as the usage environment that affects the system behavior.

In most cases, context is only roughly analyzed in comparison to functional or non-functional system requirements. As a result, unexpected behavior may emerge in a system if a developer does not recognize any possible conflicting combinations between the system and its context. It is also difficult to decide the boundary of the context that should be taken into account: which context element, an object existing outside of the system, should be included as the targets of requirements analysis.

If a developer explores the context boundary in an ad-hoc manner, he or she will be faced with the *frame problem* [14] because there are unlimited context elements in the real world where the system exists. The *frame problem* is the problem of representing the effects of the system behavior in logic without explicitly specifying a large number of conditions not affected by the behavior.

To relax the *frame problem* in embedded systems, we propose CAMEmb (Context Analysis Method for Embedded systems), a context-dependent requirements analysis method. A context model is constructed from the initial system requirements by using the *UML Profile for Context Analysis*. This context model clarifies the relation between a system and its context. In CAMEmb, only the *value-context elements*, a kind of value objects, are extracted as the associated context elements because many embedded systems are abstracted as a sensor-and-actuator machine exchanging the physical value between a system and its context. Applying the *Guide Words for Context Analysis*, we can explore only a sequence of context elements directly or indirectly affecting the data value observed or controlled by the system sensors and actuators. Other context elements not affecting the system observation and control are not taken into account because these context elements do not affect the system behavior. We can relax the *frame problem* because we only have to consider limited number of context elements as the context of the target system.

The remainder of this paper is structured as follows. In Section 2, problems in the current requirements analysis methods are pointed out in terms of the *frame problem*. In Section 3 and 4, CAMEmb is introduced to relax the *frame problem*. In Section 5, we discuss how to apply our idea to other domains such as security. In Section 6, we introduce related work. Concluding remarks are provided in Section 7.

2 Motivation

In this section, typical problems in the current requirements analysis methods are pointed out by describing the specification of an electric pot as an example.

2.1 Motivating Example

An electric pot is an embedded system for boiling water. Here, for simplicity, only the following is considered: 1) the pot has three hardware components: a heater, a thermostat, and a water level sensor; 2) the pot controls the water temperature by turning on or off the heater; 3) the pot changes its mode from the heating mode to the retaining mode when the temperature becomes 100 Celsius; and 4) the pot observes the volume from the water level sensor that detects whether water is below or above a certain base level.

In case of the electric pot, the water temperature should be taken into account as an important context element. Here, as an example, let us consider the specification that controls the water temperature. In most cases, this specification is described by implicitly taking into account the specific context—for example, such the context that water is boiled under the normal air pressure. A developer describes the software logic corresponding to the specific context—in this case, the pot continues to turn on a heater switch until the water temperature becomes 100 Celsius. Below is the specification described in pseudo code. This function describes that a controller continues to turn on a heater while the value of the temperature obtained from a thermostat is below 100 Celsius. The `Boil` function behaves correctly under the normal circumstance.

```
// Boil function
while thermostat.GetTemperature() < 100.0
  do heater.On();
```

Although this traditional approach is effective, there is room for improvements because it does not explicitly consider the context elements such as water and air pressure. The above `Boil` specification looks correct. However, faults may occur if the expected context is changed—for example, the circumstance of the low air pressure. Because the boiling point is below 100 Celsius under this circumstance, the software controller continues to heat water even if its temperature becomes the boiling point. As a result, water evaporates and finally its volume will be empty. The water level sensor observes the volume, and the pot stops heating. Although this behavior satisfies the above system specification, the pot may be useless for the people who use it up on high mountains where the air pressure is low.

2.2 Problems to be Tackled

The boundary of the context should be determined from stakeholders' requirements. If we consider climbers as customers of the pot, we have to admit that we failed in eliciting requirements in the above example.

It is not easy to define the context boundary even if the target users of the system are determined. A developer will be faced with the *frame problem* because there are unlimited context elements in the real world. There are some studies that take into account the real world as a modeling target. For example, Greenspan, S. et al. claim the necessity of introducing real world knowledge into requirement specifications [4]. But, current requirements elicitation methods do not answer a question: how and why do we find air pressure as a context element ? Of course, domain knowledge and past experiences are important to find this kind of requirements elicitation. Moreover, we admit that there are no complete methods to overcome the *frame problem*. However, at the same time, we need a method for systematically exploring the context boundary because many incidents that occur in the real embedded systems are caused by insufficient context analysis. That is, unexpected context influence that cannot be predicted in the requirements elicitation phase tends to cause a crucial incident. Many engineers in the industry face this problem.

3 CAMEmb

CAMEmb is a context analysis method for dealing with the problem pointed out in Section 2.

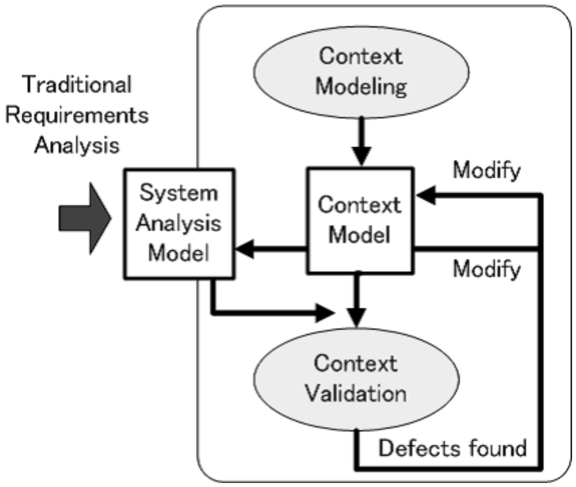


Fig. 1. CAMEmb overview

3.1 Overview

CAMEmb complements the insufficiency of the traditional requirements analysis methods as illustrated in Figure 1.

CAMEmb in the requirements analysis phase consists of 1) context modeling and 2) context validation. After traditional requirements analysis is performed from the viewpoint of eliciting the system functions and non-functional properties, CAMEmb is applied. In 1), the context elements affecting the system behavior are extracted. The boundary of the context that should be taken into account is explored. In 2), the consistency and correctness of a context model is verified using VDM++ [3], an object-oriented extension of VDM-SL (The Vienna Development Method – Specification Language). We can check whether a system analysis model behaves correctly within the expected context boundary. When the system analysis model does not behave correctly, we regard this result as the requirements elicitation defects. The context boundary is not correct or the system requirements are not feasible in the expected context. In the former case, we have to modify the context model. Otherwise, in the latter case, we have to modify the system analysis model. It depends on stakeholders' needs whether we have to modify the system analysis model or the context model.

In this paper, we focus on the context modeling method and explain its process step by step.

3.2 Context Analysis Model

Figure 2 illustrates the result of context analysis for an electric pot. The upper side and the lower side show a system and its context, respectively. The details of the *Controller* in the context model are described in the system analysis model.

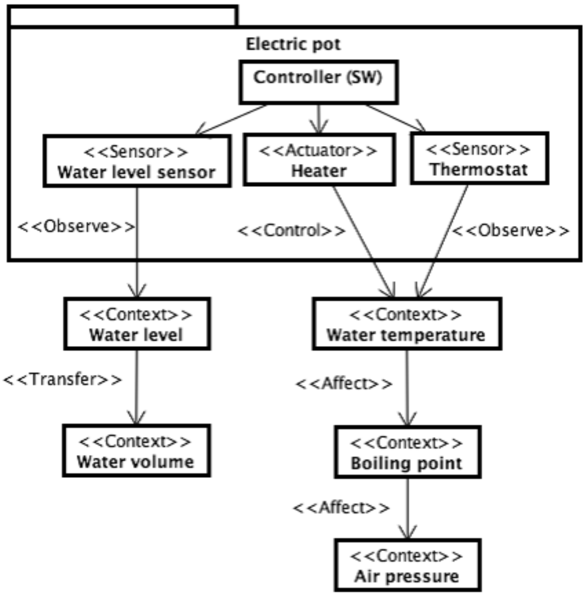


Fig. 2. Context analysis model for an electric pot

Sensors and actuators for observing or controlling the context are regarded as the interface components that separate the context from a system. Figure 2 shows only the structural aspect of the context modeling. The details of the *Controller* and the behavioral aspect of the context model are omitted due to the space limitation. In CAMEmb, the behavioral aspect is modeled using state machine diagrams. The structural aspect plays an important role in exploring the context boundary as mentioned below.

3.3 UML Profile for Context Analysis

A UML profile is provided for context analysis as shown in Table 1.

This profile can describe system elements, context elements, and associations between them: four kinds of stereotypes including $\ll Context \gg$, $\ll Hardware \gg$, $\ll Sensor \gg$, and $\ll Actuator \gg$ are defined as an extension of the UML class ($\ll Sensor \gg$ and $\ll Actuator \gg$ are subtypes of $\ll Hardware \gg$); and five kinds of stereotypes including $\ll Observe \gg$, $\ll Control \gg$, $\ll Transfer \gg$, $\ll Affect \gg$, and $\ll Noise \gg$ are defined as an extension of the UML association. The arrow of $\ll Observe \gg$ and $\ll Control \gg$ indicates the target of observation and control. The arrow of $\ll Noise \gg$ and $\ll Affect \gg$ indicates the source of noise and affect, respectively. The arrow of $\ll Transfer \gg$ indicates the source of transformation. $\ll Transfer \gg$ is introduced in order to represent data transformation because a sensor cannot

Table 1. UML profile for context analysis

Name	Category	Definition
≪ *Context* ≫	Class	Context
≪ *Hardware* ≫	Class	Hardware
≪ *Sensor* ≫	Class	Sensor (subtype of ≪ *Hardware* ≫)
≪ *Actuator* ≫	Class	Actuator (subtype of ≪ *Hardware* ≫)
≪ *Observe* ≫	Association	Sensor observes a context element
≪ *Control* ≫	Association	Actuator controls a context element
≪ *Transfer* ≫	Association	Data is transformed into a different form because a sensor cannot directly observe the original data
≪ *Affect* ≫	Association	Data from the target context element is affected by other context elements
≪ *Noise* ≫	Association	Noise from other context elements (subtype of ≪ *Affect* ≫)

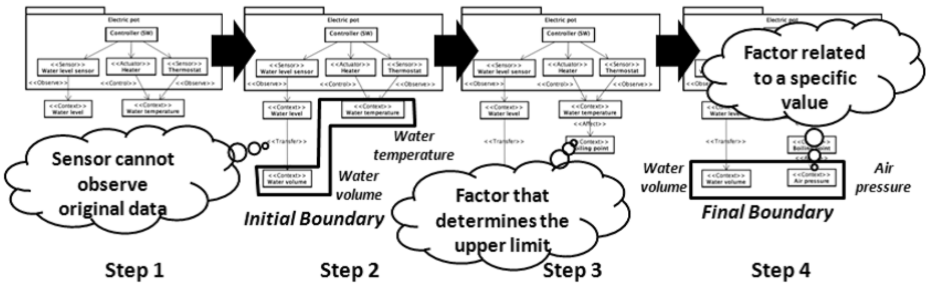

Sensor cannot observe original data
Water temperature
Water volume
Initial Boundary
Factor that determines the upper limit
Factor related to a specific value
Water volume
Air pressure
Final Boundary
Step 1
Step 2
Step 3
Step 4

Fig. 3. Stepwise context analysis using guide words (for illustration only)

directly observe the original data. For example, *water level* observed by the water level sensor is not the final observation target of a pot. That is, a pot wants to observe not the *water level* but the *water volume*.

The associations between *Controller* and three hardware components (sensors and actuators) indicate the phenomena such as *sending a command from software to hardware* and *receiving data from hardware*. However, stereotypes for these phenomena are not provided in our UML profile because they should be considered in system analysis not in context analysis.

4 Stepwise Context Analysis Using Guide Words

The context model shown in Figure 2 is created as illustrated in Figure 3. Figure 3 shows only the image of context analysis procedures. Please refer to Figure 2 when a detailed analysis result is needed.

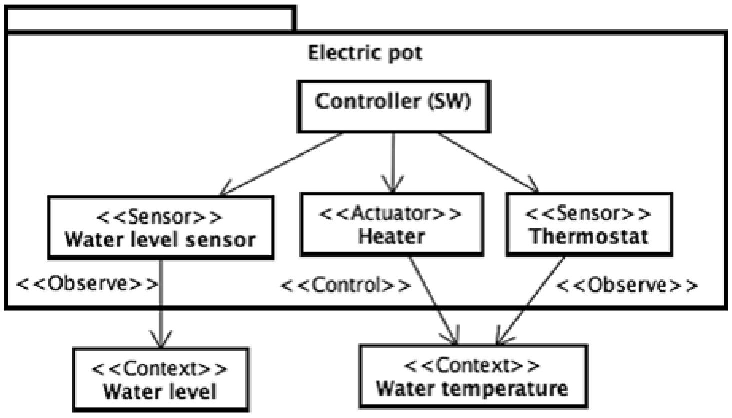


Fig. 4. Step1: extract directly observed or controlled context elements

Step1: Extract Directly Observed or Controlled Context Elements

First, context elements ($\ll$ *Context* $\gg$), which are directly observed or controlled by a sensor or an actuator, are extracted as illustrated in Figure 4.

We regard the environment value as a context element because CAMEmb focuses on embedded systems based on sensing and actuating. We call these context elements *"value-context elements"*. In case of an electric pot, *water level* and *water temperature* are extracted since *water level* is observed by the water level sensor and *water temperature* is controlled by the heater.

Step 2 [Initial Boundary]: Extract Indirectly Observed or Controlled Context Elements

An element directly observed by a sensor may be an alternative context element in such a case that the sensor cannot observe the original value of the target context element. For example, the pot wants to observe not the *water level* but the *water volume.*

Next, we explore the target context elements by using $\ll Transfer \gg$. In the step 2, all paths from sensors and actuators to the target context elements are completely extracted as illustrated in Figure 5. The initial context boundary is determined in this stage. In case of an electric pot, *water volume* and *water temperature* are extracted as the initial context boundary.

Step 3 [Intermediate Boundary]: Extract Impact Factors Using Guide Words

The initial context boundary is an ideal boundary in which system's sensing and controlling are not affected by other factors. However, there are many factors affecting observation and actuation in the real world. We have to extract these factors in order to develop reliable embedded systems.

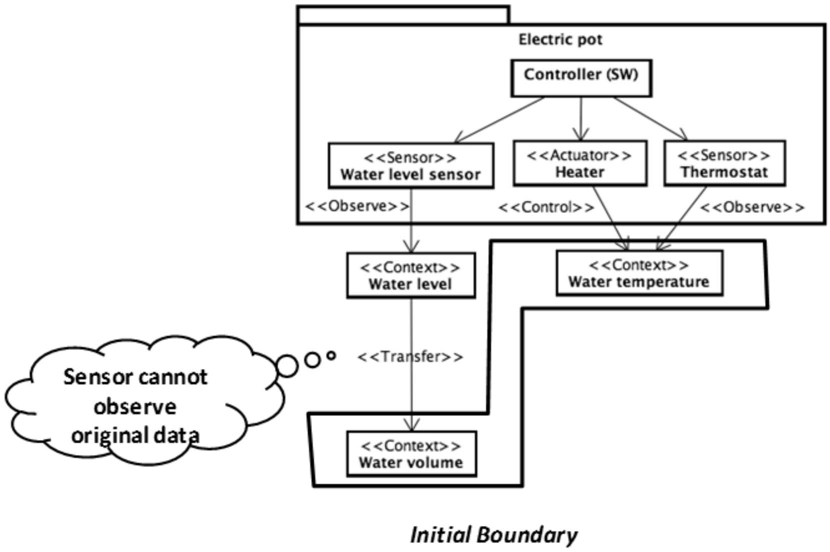


Fig. 5. Step 2 [Initial boundary]: extract indirectly observed or controlled context elements

Table 2. Guide words for context analysis

No.	Category of $\ll Affect \gg$	Guide word
1.	physical phenomena	factor that determines the upper limit
2.	physical phenomena	factor that determines the lower limit
3.	physical phenomena	factor related to a specific value
4.	influence to sensing	factor that interferes with the observation
5.	influence to actuation	factor that interferes with the control

In CAMEmb, impact factors that affect the states of these context elements are extracted using guide words. Guide words, hints for deriving related elements, are effective for software deviation analysis [13]. Guide words are mainly used in HAZOP (Hazard and Operability Studies). In HAZOP, deviation analysis is performed by using the guide words including *NOT*, *MORE*, *LESS*, *AS WELL AS*, *PART OF*, *REVERSE*, and *OTHER THAN*. For example, *higher pressure*, which may be deviated from a normal situation, can be derived from the property *pressure* and the guide word *high*.

In addition to the HAZOP guide words, CAMEmb provides a set of guide words specific to the context analysis as shown in Table 2. These guide words help us to find an obstacle that affects the system observation and control in terms of the *context-value*. By using these guide words, we can extract context elements that affect the context elements existing within the initial boundary. Our guide words can be considered as hints for deviation analysis targeted to context analysis. If there is a context element having the influence on another context element, we link them by the $\ll Affect \gg$ association.

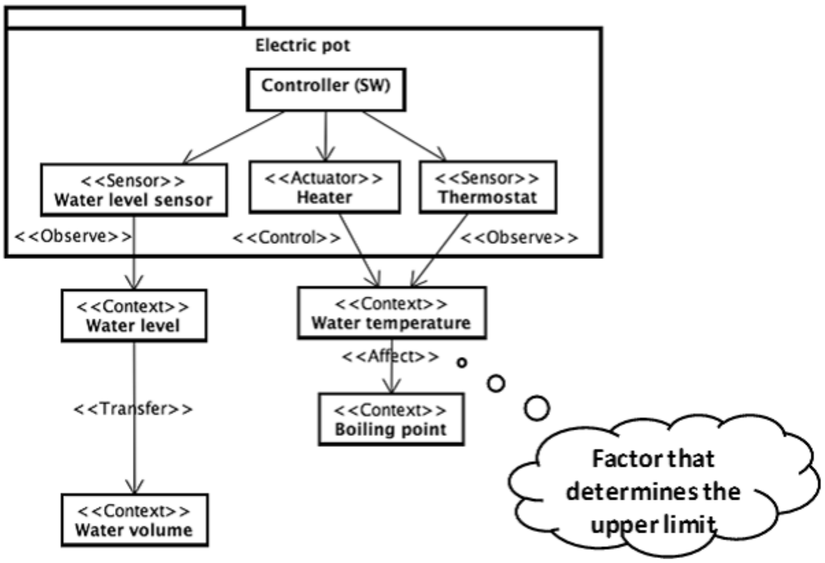


Fig. 6. Step 3 [intermediate boundary]: extract impact factors using guide words

In case of an electric pot, the *boiling point* can be extracted as an impact factor for the *water temperature* by applying the guide word *"factor that determines the upper limit"* since the temperature does not become higher than the boiling point. This guide word indicates that we have to take into account the *boiling point* when we develop an embedded product controlling the *water temperature*. Figure 6 shows this stage of the context analysis. Without guide words, we have to explore impact factors in an ad-hoc manner and we may not be able to find any impact factors.

Step 4 [Final Boundary]: Determine the Context Boundary

We have to continue to extract impact factors as many as possible to develop reliable systems. In case of an electric pot, the *air pressure* can be extracted as an impact factor for the *boiling point* by applying the guide word *"factor related to a specific value"* since the boiling point of the water is 100 Celsius under the circumstance of 1.0 atm. At this point, we finish the context exploration because we can find no more impact factors affecting the *air pressure*. Figure 7 shows this stage of the context analysis. We can extract two context elements *water volume* and *air pressure* as the final context boundary.

It depends on the domain knowledge or experience of a development organization when a developer stops exploring related value-contexts elements. Our method helps a developer to extract context elements affecting the system behavior as many as possible using domain knowledge inspired by guide words. Without domain knowledge, it is not necessarily easy to find impact factors only using guide words. For example, a developer, who does not have enough knowledge about

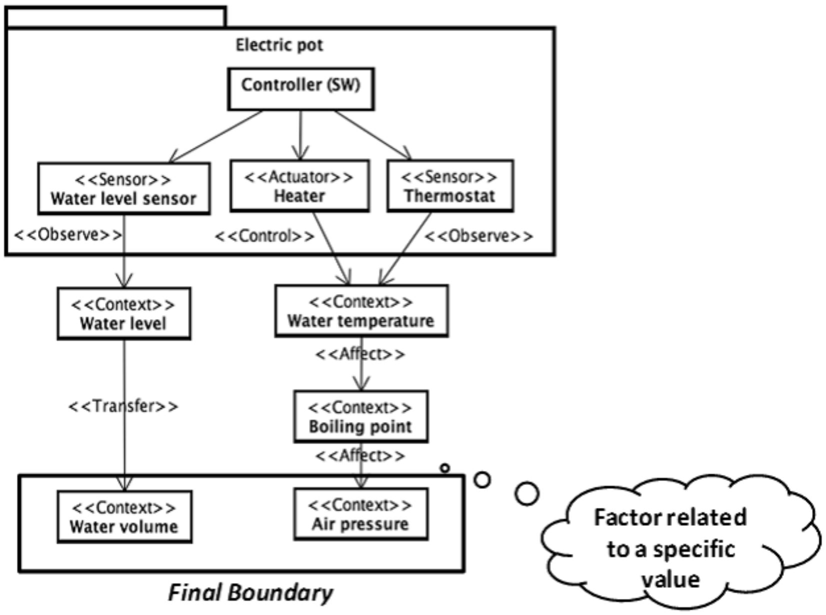


Fig. 7. Step 4 [Final boundary]: determine the context boundary

physics, may not be able to find the *air pressure* even if the developer knows the guide word *"factor related to a specific value"*. Domain knowledge plays an important role in context modeling. However, it is insufficient unless a systematic way for extracting domain knowledge exists. Our guide words can derive domain knowledge needed for context-based deviation analysis from a developer.

As shown here, the boundary of the context is explored by using *UML Profile for Context Analysis* and *Guide Words for Context Analysis*. We can explore only a sequence of context elements directly or indirectly affecting the data value observed or controlled by the system sensors and actuators. Other context elements not affecting the system observation and control are not extracted. There are many context elements such as person, table, and light in the environment of an electric pot. However, these context elements do not affect the data observed or controlled by the pot. So, we do not have to take into account these context elements. These context elements exist out of the boundary.

5 Discussion

In this section, we discuss on the applicability of CAMEmb.

5.1 Applicability

We examined the proposed method to another embedded system: a line trace car. The car runs tracing a line by observing a ground color.

Table 3. Applicability of guide words

No.	Category of $\ll Affect \gg$	Guide word	Example	Case study
1.	physical phenomena	factor that determines the upper limit	boiling point	electric pot
2.	physical phenomena	factor that determines the lower limit	freezing point	electric pot
3.	physical phenomena	factor related to a specific value	air pressure	electric pot
4.	influence to sensing	factor that interferes with the observation	light	line trace car
5.	influence to actuation	factor that interferes with the control	light	line trace car

In CAMEmb, *guide words* play a important role to find context elements affecting the system behavior. Table 3 shows how guide words are applied to two case studies: electric pot and line trace car. Although both of two case studies are sensor-actuator systems, their characteristics are different. In an electric pot, its system behavior is affected by physical environment in which the pot exists. On the other hand, in a line trace car, its system behavior is affected by obstacles of sensing and actuating. There are three kinds of $\ll Affect \gg$: *physical phenomena*, *influence to sensing*, and *influence to actuation*. The guide words related to *physical phenomena* are used in the electric pot. The guide words related to *influence to sensing* and *influence to actuation* are used in the line trace car.

As mentioned here, we can apply our approach to two kinds of sensor-actuator systems.

Although we may not be able to apply CAMEmb to all the application domains, there are many domains that can be modelled as monitor-controller (or sensor-actuator) systems. Security, safety, network threats, and user interactions are examples of such domains. In these domains, context can be analyzed using our approach. For example, *trust* in the security domains correspond to *value* in CAMEmb. By defining the *guide words* that affect the trusts, we can explore the trust boundary.

5.2 Avoidance of the Frame Problem

In CAMEmb, we select only the elements affecting the data value observed or controlled by a system. We think that the value-based context analysis is reasonable because most embedded systems observe the input data from the environment through sensors and affect the environment by emitting the physical outputs through actuators. The system behavior is determined by the data observed by the sensors and controlled by the actuators. We have only to take into account the context elements explicitly or implicitly affecting the data linked with the $\ll Transfer \gg$ or the $\ll Affect \gg$ associations. The context analysis terminates when there are no more context elements affecting the data. In our approach, the affection is determined by using guide words. Of course, the method using guide words is not complete. But, the method helps a developer to find the context elements affecting the system behavior as many as possible.

6 Related Work

Jackson, M. proposes the *problem frames approach* [9] in which relations between a machine (a system to be developed) and the real world are explicitly described. The *problem frames approach* emphasises on the importance of analysing the real world and the problems.

First, in this section, we discuss on the relation between CAMEmb and the *problem frame approach*. There are several common ideas between them. We believe that CAMEmb provides a fruitful mechanism for using the *problem frames approach* more effectively. The *problem frames approach* is strong in analysing the real world (context) in terms of the problems. On the other hand, CAMEmb is strong in exploring the context boundary and refining a context model to the corresponding software design model.

Next, we show the other related work.

6.1 Problem Frames

A context diagram in the *problem frames approach* describes problem domains in an application domain, their connections, and a machine and its connections to the problem domains. The notion of context in CAMEmb corresponds to the real world in the problem frame. Examples of formalising requirements with problem frames can be found in [2][6].

We are now exploring the possibility of integrating CAMEmb with the *problem frames approach*. Figure 8 shows a context analysis model described in the *problem frames approach*. We can describe the context diagram of a line trace car by using the *Required Behavior* frame and the *Transformation* frame. There is the similarity between our UML profile and frame patterns. For example, $\ll Transfer \gg$ corresponds to *Transformation* frame.

We consider that it is effective to apply CAMEmb after problem analysis is done. The *problem frames approach* is strong in analysing problems in the real world. On the other hand, our approach provides a systematic way for determining the context boundary. Because a context analysis model in the *problem frames approach* and that in CAMEmb can be converted each other as shown in Figure 8, we believe that both methods can be integrated.

6.2 Four-Variable Model

Parnas, D. L. and Madey J. propose the *four-variable model* [15], a model for system requirements and design. This model takes into account environmental quantities such as physical proprieties, values displayed on devices, and states of controlled devices. In the *four-variable model*, functions, timing, and correctness are described by using monitored variables, control variable, and input / output data items because variables internal to the system are not adequate for specifying system requirements. The *four-variable model* was used to specify the requirements for the A-7 aircraft in SCR (Software Cost Reduction) [7][8] providing a tabular notation for specifying requirements.

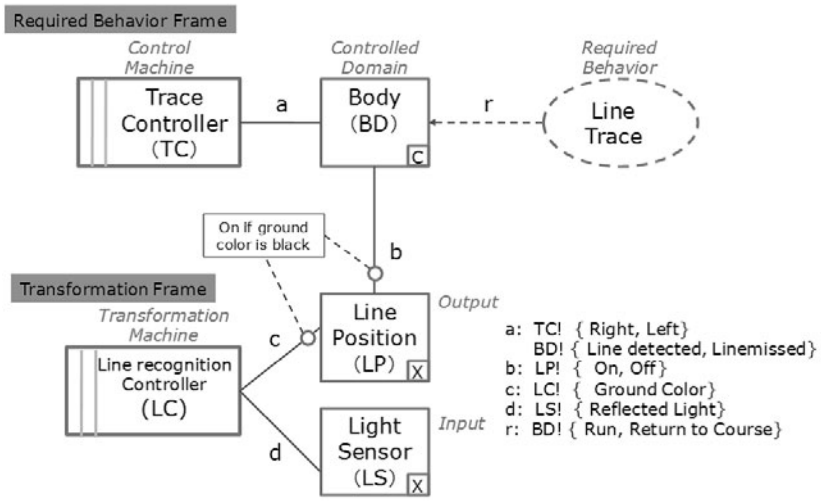


Fig. 8. Context analysis model described in problem frames

The *four-variable model* is similar to CAMEmb because monitored variables and control variables correspond to context elements observed by sensors and controlled by actuators. However, the *four-variable model* does not provide a way for finding context elements and exploring the context boundary. CAMEmb focuses on the recognition paths affecting the system observation and control.

6.3 Context Analysis and SPL

Software product line (SPL) [11] is a promising approach to developing embedded systems. A product is constructed by assembling core assets, components reused in a family of products. In the current SPL, however, the feature analysis is mainly conducted from the viewpoint of system configurations and context is not considered explicitly. However, some PLE methods provide a way for modeling the context of a product [1][16][5][17][12].

Atkinson, C. et al. propose a PLE method called KobrA [1] in which the context realization models are described by analyzing contexts of target systems. However, the systems and contexts are simultaneously described in KobrA. On the other hand, the system lines are completely separated from the context lines in our approach. We believe that our approach is effective comparing to the way in which contexts are taken into account as one of the system concerns. There are contexts features that can be shared among multiple system lines. If a context belongs to a specific system line, the context cannot be reused in other system lines.

Kang, K. C. et al. propose a method for categorizing features into four layer including capability, operating environment, domain technology, and implementation technique [10]. They point out the importance of introducing the viewpoint of *usage context*. Lee, K. and Kang, K. C. propose a model showing how product usage contexts are related to product features [12]. In the model, physical contexts indicate physical environments or locations where a product is deployed and used.

Reiser, M. O. et al. use the concept of a product set to describe contextual constraints for feature election. Hartman, H. and Trew, T. propose a context variability model (CVM) [5] for modeling product lines that support several dimensions in the context space. Their approach is similar to our approach. In CVM, a separate feature diagram is used to model contextual variability and linked to a product line feature diagram. Similarly, Tun et al. describe contextual variability as a separate feature diagram [17]. The work of Tun is also based on the work of Jackson's problem frames.

We previously proposed a context-dependent SPL method [19] in which a product line was composed of system and context lines. The former is obtained by analyzing a family of systems. The latter is obtained by analyzing the features of the expected context. The configuration of selected system components and context elements can be formally checked by using VDMTools. Historically, the prototype of CAMEmb was originally proposed as a method for constructing context lines. After that, we became aware that our approach could be generalized and could be applied to not only SPL but also requirements elicitation. Moreover, we became aware that the idea of *value-context elements* could play an important role to relax the *frame problem*. In this paper, we positioned CAMEmb as a requirements elicitation method for exploring the context boundary.

7 Conclusion

In this paper, we proposed CAMEmb, a context-dependent requirements analysis method. As demonstrated in this paper, we could provide a method for exploring the context boundary. The idea of *value-context elements* and *guide words* plays an important role.

It is favorable if a system analysis model and a context analysis model are transformed into a design model reflecting the contexts. We have developed a prototype of a domain-specific modeling environment for supporting context analysis [18]. This tool consists of a model editor for supporting a *UML Profile for Context Analysis*, a model compiler for transforming a system analysis model and a context model to a system design model, and a code generator from the system design model to Java. The generated design model consists of three layers *Controller*, *Context Recognition*, and *Driver*. The results of context analysis can be reflected to the software architecture of the target system. The *Context Recognition* layer, the most important part in the design model, is obtained by $\ll Transfer \gg$ and $\ll Affect \gg$ relations in the context model.

We plan to develop a method for integrating CAMEmb with the *problem frames approach*. The former is strong in exploring the context boundary and the latter is strong in analysing the real world. Tool support is necessary to accomplish this research goal. For example, translation between a context model described in CAMEmb and a context diagram in the *problem frames approach* should be supported. It would be better if problem analysis, context analysis, and refinement to a design model are seamlessly linked.

We think that the essential idea of CAMEmb can be applied to other kinds of context such as security and safety in embedded systems. As the next step, we plan to apply CAMEmb to such an application.

Acknowledgement. This research is being conducted as a part of the Grant-in-aid for Scientific Research (B), 23300010, 2011 by the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

1. Atkinson, C., et al.: Component-Based Product Line Engineering with the UML. Addison-Wesley (2001)
2. Coleman, J.W., Jones, C.B.: Examples of how to Determine the Specifications of Control Systems. In: Proceedings of Workshop on Rigorous Engineering of Fault-Tolerant Systems (REFT 2005), pp. 65–73 (2005)
3. Fitzgerald, J., Larsen, G.P., Mukherjee, P., Plat, N., Verhoef, M.: Validated Designs for Object-oriented Systems. Springer (2005)
4. Greenspan, S., Mylopoulos, J., Borgida, A.: Capturing More World Knowledge in the Requirements Specification. In: Proceedings of International Conference on Software Engineering (ICSE 1982), pp. 225–234 (1982)
5. Hartmann, H., Trew, T.: Using Feature Diagrams with Context Variability to Model Multiple Product Lines for Software Supply Chains. In: Proceedings of the 12th International Software Product Line Conference (SPLC 2008), pp. 12–21 (2008)
6. Hayes, I., Jackson, M., Jones, C.: Determining the Specification of a Control System from That of Its Environment. In: Araki, K., Gnesi, S., Mandrioli, D. (eds.) FME 2003. LNCS, vol. 2805, pp. 154–169. Springer, Heidelberg (2003)
7. Heitmeyer, C.L., Bull, A., Gasarch, C., Labaw, B.G.: SCR*: A Toolset for Specifying and Analyzing Requirements. In: Proceedings of Computer Assurance (COMPASS), pp. 109–122 (1995)
8. Heitmeyer, C., Bharadwaj, R.: Applying the SCR Requirements Method to the Light Control Case Study. Journal of Universal Computer Science 6(7), 650–678 (2000)
9. Jackson, M.: Problem Frame: Analyzing and Structuring Software Development Problems. Addison-Wesley (2001)
10. Kang, K.C., Kim, S., Lee, J., Shin, E., Huh, M.: FORM: A Feature-oriented Reuse Method with Domain-specific Reference Architecture. Annals of Software Engineering 5, 143–168 (1998)
11. Kang, K.C., Lee, J., Donohoe, P.: Feature-Oriented Product Line Engineering. IEEE Software 9(4), 58–65 (2002)
12. Lee, K., Kang, K.C.: Usage Context as Key Driver for Feature Selection. In: Bosch, J., Lee, J. (eds.) SPLC 2010. LNCS, vol. 6287, pp. 32–46. Springer, Heidelberg (2010)
13. Leveson, N.G.: Safeware: System Safety and Computers. Addison-Wesley Publishing Company (1995)
14. McCarthy, J., Hayes, P.J.: Some Philosophical Problems from the Standpoint of Artificial Intelligence. Machine Intelligence 4, 463–502 (1969)
15. Parnas, D.L., Madey, J.: Functional Documentation for Computer Systems Engineering, McMaster University, Technical Report CRL 237 (1991)

16. Reiser, M.-O., Weber, M.: Using Product Sets to Define Complex Product Decisions. In: Obbink, H., Pohl, K. (eds.) SPLC 2005. LNCS, vol. 3714, pp. 21–32. Springer, Heidelberg (2005)
17. Tun, T.T., Boucher, Q., Classen, A., Hubaux, A., Heymans, P.: Relating Requirements and Features Configurations: A Systematic Approach. In: Proceedings of the 13th International Software Product Line Conference (SPLC 2009), pp. 201–210 (2009)
18. Ubayashi, N., Otsubo, G., Noda, K., Yoshida, J.: An Extensible Aspect-Oriented Modeling Environment. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 17–31. Springer, Heidelberg (2009)
19. Ubayashi, N., Nakajima, S., Hirayama, M.: Context-Dependent Product Line Practice for Constructing Reliable Embedded Systems. In: Bosch, J., Lee, J. (eds.) SPLC 2010. LNCS, vol. 6287, pp. 1–15. Springer, Heidelberg (2010)

Author Index

Abramov, Jenny 126
Anson, Omer 126

Bellahsene, Zohra 79
Bergmann, Gábor 110
Bose, R.P. Jagadeesh Chandra 33, 202
Boussaid, Omar 172
Buschle, Markus 1
Businska, Ligita 186

Coletta, Remi 79

Ekstedt, Mathias 1

Falcão e Cunha, João 64

Galvao, Teresa 64
Grigera, Julián 142

Holm, Hannes 1

Islam, Shareeful 95

Kamei, Yasutaka 218
Kirikova, Marite 186
Koch, Nora 142

Levín, Ana M. 49
Loudcher, Sabine 172

Massacci, Fabio 110
Mouratidis, Haralambos 95
Mourinho, Joao 64

Ngo, DuyHoa 79

Pacheco, Jose 64
Paci, Federica 110
Pastor Lopez, Oscar 49
Pavlidis, Michalis 95
Pinggera, Jakob 16

Reinhartz-Berger, Iris 156
Rivero, José Matías 142
Robles Luna, Esteban 142
Rossi, Gustavo 142

Shahzad, Khurram 1
Shoval, Peretz 126
Sommestad, Teodor 1
Sturm, Arnon 126, 156

Tsoury, Arava 156
Tun, Thein Than 110

Ubayashi, Naoyasu 218

Valverde, Francisco 49
van der Aalst, Wil M.P. 33, 202
Varró, Dániel 110
Verbeek, Eric H.M.W. 33
Vieira, Fernando 64
Villanueva, Maria José 49

Weber, Barbara 16

Yu, Yijun 110

Zugal, Stefan 16