CAiSE '15

27th International Conference on Advanced Information Systems Engineering

8-12 June 2015, Stockholm, Sweden http://caise2015.dsv.su.se

Stockholm University

caise.conference

Linkedin caise-conference

@CAiSEConference

Jānis Grabis
Kurt Sandkuhl
(Eds.)

CAiSE Forum 2015

Proceedings of the CAiSE'15 Forum
at the 27th International Conference on
Advanced Information Systems Engineering

CAiSE 2015

June 8-12, 2015
Stockholm, Sweden

Title
Proceedings of CAiSE Forum 2015

Sub-title
27th International Conference on Advanced Information Systems Engineering
CAiSE 2015
Stockholm, Sweden, June 8-12, 2015

Volume Editors
Jānis Grabis
Riga Technical University
Department of Management Information Technology
Meža iela 1/3
Rīga LV-1048, Latvia
E-mail: grabis@iti.rtu.lv

Kurt Sandkuhl
University of Rostock
Institute of Computer Science
Albert-Einstein-Str. 22
D-18 057 Rostock, Germany
E-mail: kurt.sandkuhl@uni-rostock.de

Preface

CAiSE 2015 is the 27th in the series of International Conferences on Advanced Information System Engineering. The theme of CAiSE 2015 is Creativity, Ability and Integrity in information Systems Engineering. Creativity implies designing Information Systems in novel and unexpected ways making use of information from different sources that need to be merged and molded to become meaningful and valuable. Ability enables systems to be capable to deliver business in excellent, competitive and agile ways. Integrity refers to ensuring security, trustworthiness and compliance with ethical code of information systems.

The CAiSE Forum is a place within the CAiSE conference for presenting and discussing new ideas and tools related to information systems engineering. Intended to serve as an interactive platform, the Forum aims at the presentation of emerging new topics, case studies as well as demonstration of innovative systems, tools and applications. It includes three types of contributions: 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; and 3) Case study reports focusing on real world cases in information systems engineering. The CAiSE Forum 2015 proceedings present a collection of 31 excellent short research papers and demos. The papers were selected by picking visionary papers out of those submitted to the CAiSE conference to stimulate open discussions of high-quality on-going research as well as by competitive review of the papers submitted directly to the CAiSE Forum.

The selection of papers spans all three of the aforementioned conference themes and includes a multitude of topics: Mobile technologies, advanced visualization, human-computer interaction and gamification leads to creative information systems engineering methods. Novel model driven development, process management and data management methods contribute to continuous improvement of our ability to develop high quality information systems. The papers devoted to the integrity topic emphasize security maintenance, fraud detection and compliance and risk management issues. Finally, the papers show strong synergies among integrity and creativity and ability. New creative information systems and digital services can be created and widely adopted only if they respect requirements of all parties involved, especially, concerning security and privacy requirements.

We would like to thank everyone who contributed to the CAiSE 2015 conference and CAiSE Forum 2015 in particular. We thank the authors for contributing and presenting their research, we appreciate invaluable contribution of the members of the Program Committee and external reviewers and we thank all members of the local organization team from the Stockholm University for ensuring smooth processing of the conference. We acknowledge the EasyChair development team for providing such a convenient tool for preparing these proceedings.

June, 2015

Jānis Grabis
Kurt Sandkuhl

CAiSE 2015 Organization

Steering Committee

Barbara Pernici, Politecnico di Milano, Italy
Oscar Pastor, Universitat Politècnica València, Spain
John Krogstie, Norwegian University of Science and Technology, Norway

Advisory Committee

Janis Bubenko Jr, Royal Institute of Technology, Sweden
Arne Sølvberg, Norwegian University of Science and Technology, Norway
Colette Rolland, University of Paris 1 – Panthéon – Sorbonne, France

General Chair

Paul Johannesson, Stockholm University, Sweden

Program Chairs

Jelena Zdravkovic, Stockholm University, Sweden
Marite Kirikova, Riga Technical University, Latvia

Organisation Chairs

Åsa Smedberg, Stockholm University, Sweden
Eric-Oluf Svee, Stockholm University, Sweden
Iyad Zikra, Stockholm University, Sweden
Birger Andersson, Stockholm University, Sweden
Parisa Aasi, Stockholm University, Sweden

Forum Chairs

Janis Grabis, Riga Technical University, Latvia
Kurt Sandkuhl, University of Rostock, Germany

CAiSE Forum 2015 Program Committee

Joao Paulo Almeida, Federal University of Espirito Santo, Brazil
Said Assar, Institut Mines-Telecom, France
Eduard Babkin, Higher School of Economics Nizhny Novgorod, Russia
Luciano Baresi, DEIB - Politecnico di Milano, Italy
Clara Bassano, "Parthenope" University of Naples, Italy
Kahina Bessai, CRI   University Paris1 Pantheon Sorbonne, France
Solvita Bērziša, Riga Technical University, Latvia
François Charoy, Université de Lorraine - LORIA – Inria, France
Fabiano Dalpiaz, Utrecht University, The Netherlands

Additional Reviewers

Table of Contents

Integrity

Trusted, Fair Multi-Segment Business Models, Enabled by a User-Centric, Privacy-Aware Platform, for a Data-Driven Era

Iosif Alvertis, Michael Petychakis, Romanos Tsouroplis, Evmorfia Biliri, Fenareti Lampathaki, Dimitrios Askounis
National Technical University of Athens
Athens,Greece
{alvertisjo, mpetyx, rtsouroplis, ebiliri, flamp, askous}@epu.ntua.gr

Timotheos Kastrinogianins, Andreas Daskalopoulos Theodoros Michalareas
VELTI S.A.
Athens, Greece
{tkastrinogiannis, adaskalopoulos, tmichalareas}@velti.com

Eric Robson, Donal MacCarthy, Christine O'Meara
TSSG, Waterford Institute of Technology
Waterford, Ireland
{ erobson, dmccarthy, comeara } @tssg.org

Lukasz Radziwonowicz , Robert Kleinfeld
Fraunhofer FOKUS,
Berlin, Germany
{ lukasz.radziwonowicz, robert.kleinfeld} @ fokus.fraunhofer.de

Abstract. Today's mobile applications, spanning from file storage and syncing, place checking in and tagging, multimedia sharing, streaming and recommendation, to social and professional networking, rely a lot on cloud-based services, which in turn offer increasingly more functionalities through their publicly available APIs. Users have greeted this abundance of services with even greater demand for innovative applications to address personal and business needs, but are still unaware of the full power that could be leveraged from their data, currently scattered on various platforms. In this complex ecosystem of mobile applications OPENi offers a way to facilitate developers into building applications over the multiple and diverse APIs but also provides users with their own personal cloud storage space, the "Cloudlet", equipped with strong privacy controlling mechanisms, accessed through its open-source Graph API platform. New business models can thus be deployed that leverage the full potential of the available data and metadata, offering application developers the means to create powerful but also privacy-aware services, respecting the requirements of both parties, i.e. service providers and data-owners.
Keywords: APIs, Cloud-based Mobile Applications, Generic Graph API, Context, Personal Data Privacy, Privacy-by-design.

1. Introduction

A new data economy has started to emerge that makes aggregated data valuable, and the conflict is around who owns personal data and how businesses can process or build additional value over this information. Data mining and analytics advancements provide more revolutionary insights than ever to understand and even predict where humans focus their needs, attention and activity at the individual, network and global level. According to the World Economic Forum in 2011 [21], personal data is becoming the new “oil”, a valuable resource of the 21st century that “will touch all aspects of society”. Under this economy companies act as data brokers and get paid for data sets with or without users’ permission; for example, the US data brokerage industry is worth in the region of $15bn [6]. But this is not the first time doing such activities, traditional brokers exchange customer data, like contact details and demographics for decades, used in phone marketing, marker research businesses, or for evaluating prospective customers for their credit risk. The players of note operate globally and include Experian[7] and Bluekai[11].

The main reasons are the rise of web 2.0 applications and then the dominance of smartphone applications and cloud services, which have given boost to multi-segment business models evolved since the web 1.0 era; users’ personal data coming from free-of-charge services are offered through advertising services to business customers who wanted to sell targeted advertisements in the social media age. Nevertheless, there is great difference since web 1.0 services: personal data is fragmented, stored across various services, in walled silos, exposed under proprietary APIs and not available on the broader Internet via web standards. Thus companies like Google and Amazon continue their traditional businesses, but the last decade belongs to companies like Facebook and Twitter which do not contribute on the Web but ask to build mobile, web and desktop applications through their APIs, which keep information fragmented, unrelated and controlled under strict policies that change once a while based on the strategy of the companies.

Nevertheless, new business models addressing the needs and worries of users, as well as new policies introduces, have started changing the market related with personal data. Motivated by this emerging and constantly changing landscape, OPENi [1][2] has designed and delivered a novel consumer-centric, privacy-by-design, open source, cloud-based development platform, serving as a catalyst for new applications era. This paper aims at pointing out a new business model, where end-users own and control their data, developers build applications on a distributed authorization mechanism (i.e. no central authority, thus less business risks) and enterprises can host such services in bundled telecommunication organisations, e.g. cloud hosting provided by carriers.

In section 2 there is a detailed description of the privacy-concert business models emerging lately. Section 3 describes relative work, with other solutions in the data-privacy aware area. Section 4 describes the OPENi solution with its architecture and major components. Finally section 5 has the conclusions of that work and future research.

2. Current landscape

These new business modes and terms for building a platform for new, cloud-based, social-driven and mobile enabled applications, have moved the value away from users who create data and should own them, while companies exploited business ideas of third party developers until they grow their communities. Such approach has created (a) worried, socially overexposed and exploited users who afraid how their data is used, (b) policy makers and data privacy organisations who worry and try to control how such companies use personal data, and (c) frustrated developers who either look to develop their own community with social end-points or work to adapt in any change in enterprises' API terms of use that constantly change.

Relatively to addled consumers, a recently published PEW report [4] revealed that 91% of adults agreed or strongly agreed that consumers had lost control over personal information collected and used by companies. Moreover, 80% of those who use social networking sites say they are concerned about third parties accessing data they share on these services. For that reason customers start turning into new privacy respectful tools, such as the Epic browser [9] or DuckDuckGo [10] search engine. However, individuals are often willing to compromise on privacy in return for benefits; 55% of respondents in the aforementioned PEW research indicated that they would be willing to share some information about themselves with companies in order to use online services for free. Other pain-points experienced by individuals on their online presence include headaches associated with digital identity and asset management and generally online fraud, cybercrime and identity theft.

From the side of active regulators, EU-based data regulation is about to be released and for the first time all member states will be bound by a common data privacy policy. Some aspects include the requirement of consent, required high levels of transparency with explicit communication on all uses/sharing of data, the need to keep data within EU jurisdiction, the right to be forgotten and some hefty fines for non-compliance (up to 2% of annual turnover). At US governmental level, Smart Disclosure[12] is a policy initiative designed to help individuals access personal information in formats they can use and make better decisions based on them.

Table 1. Forces changing typical business models dealing with personal data

Addled Consumer	Active Regulator	Opportunistic Enterprise
• Fraud fear • Privacy concerns • Value aware • Digital identity headaches	• Imminent EU regulation • US FTC recommendations	• New services/business models • Enterprise cost to manage data • Leveraging the data asset • External pressures

This shift in individual and public level around data privacy has woken up global players, who try to build on this emerging trend and build more balanced, compliant with the policies business solutions [15]; estimations are that there is a new player every week in this area [3]. Such services can be categorised according to several types of proposition: (a) Storage & Utility, (b) Transparency & Trust, (c) Marketing & CRM Tools and (d) New World Data Traders. However, the market is still fragmented and there is no clear market leader among the market entrants, as organisations have to solve the multiple challenges including disparate views of the individuals and legacy platform integration challenges.

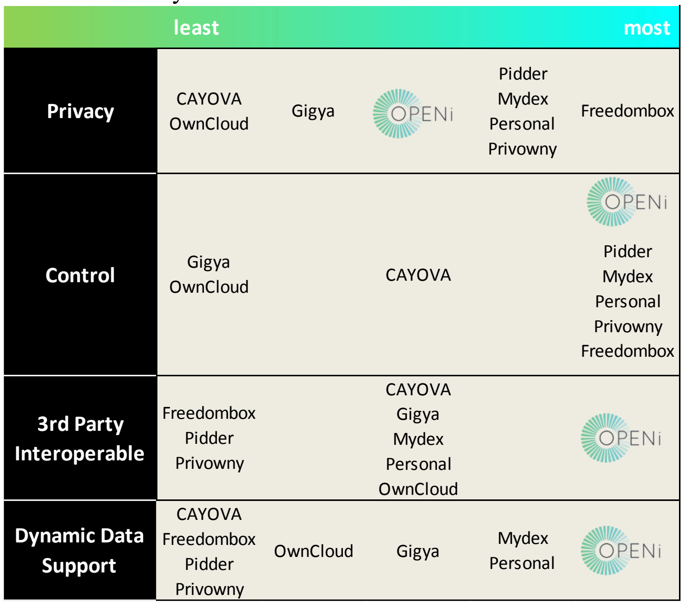
Thus, legacy business models may be under threat by new, innovative propositions [8]. The 2013 UK MiiData Pilot[5] in this area identified some critical factors for adoption, which are: (a) data must flow, thus incorporate multiple compliant sources, (b) consumer participation is essential, (c) needs-driven approach is better than data-driven focus at early market stage, (d) the proposition should focus on value, not on data, (e) convenience is important, (f) customer control is a benefit in its own right, and (g) trust and safety are of fundamental importance.

3. Relative Work

In this section an analysis of the personal data storage solutions and businesses takes place. Most of the reviewed systems are commercial platforms and some are open source solutions. In general they are middle-ware systems (so called Backend-as-a-Service) or cloud data storages which abstract external APIs and add scalability, management capabilities and other value-adding services to them. The systems act as a proxy and can rewrite request and response formats to create an interoperability layer between the system and the back-end APIs.

Cayova is an emerging social networking site set up to empower its users in the context of data privacy and control. In their privacy policy it states there are no assurances as to the security of those data but that they do their utmost to protect them. All interactions exploit HTTPS to protect users. Transactions are also protected using SSL. One the other hand, FreedomBox chooses a different approach by having a distributed one, and it is completely open source. Gigya is another commercial company that facilitates the gathering, transfer and storage of user data. This commits to not selling or renting personally identifiable information to third parties. In a similar direction, Personal Inc, provides secure storage as a service to its consumers. Similar approaches like Mydex, OwnCloud, Pidder, Qiy or even Privowny try to accomplish the same goals with a variety of similar methods. Most of those are based on standards and open source projects. What is more, in the analysis provided by Abbas [18] and that of Rahimi [19] we can see that they are not that far from the ones we already analysed above. Finally, a really interesting work that is worth mentioning is that of Zhu [20] that makes a considerable effort into the privacy field.

Figure 1 compares personal data storage services similar to OPENi’s under a number of criteria. The diagram shows that OPENi is grouped with the companies that allow their users the most control through its authorization mechanism that gives permissions both per object and per instance, it is the most interoperable as it operates under various standards (i.e. Schema.org, JSON LD), and it has the most support for dynamic data as it can be extended with more objects and connections through an API builder. However, the capability for providers to build new applications over a personal storage, even if the users can control what the developer can see, creates some compromises relatively to the way the developers use users' data; technologically it is not yet possible, and legally blur, to track data across various services and check how they are used.



	least				most
Privacy	CAYOVA OwnCloud	Gigya	OPENi	Pidder Mydex Personal Privowny	Freedombox
Control	Gigya OwnCloud		CAYOVA		OPENi Pidder Mydex Personal Privowny Freedombox
3rd Party Interoperable	Freedombox Pidder Privowny		CAYOVA Gigya Mydex Personal OwnCloud		OPENi
Dynamic Data Support	CAYOVA Freedombox Pidder Privowny	OwnCloud	Gigya	Mydex Personal	OPENi

Figure 1 - Personal Data Storage Services

4. Personal Cloudlets exposed under a unified API platform

OPENi aims to offer both data owners - thus web and mobile applications users - and developers the means to leverage the power of the vast amount of collected information, in a privacy-aware way. The envisioned "Cloudlet", a personal storage space with fine-grained permission mechanisms, holds a central role in this new mobile applications ecosystem.

As seen from the user perspective, the increasing usage of web services has led to a large amount of personal data, like place visits, images, purchases etc., being created and stored in various web storage spaces, each with its own privacy terms and conditions. OPENi brings all this information under the same roof, with central privacy mechanisms enforced by the data owner. To that end, after careful analysis of many popular APIs and of the data that they exchange, multiple APIs (e.g. Activity API, Media API, Location API, Product and Services API etc.) and relative objects been created to support the large variety of information being stored in the Cloudlets. Contextual information is also stored together with every OPENi object, under a dynamic Context API that extents meta-data attached in every object. The power of this data aggregation is thus revealed to its owner who can better appreciate its potential, as well as developers who can find rich semantically information to build better applications.

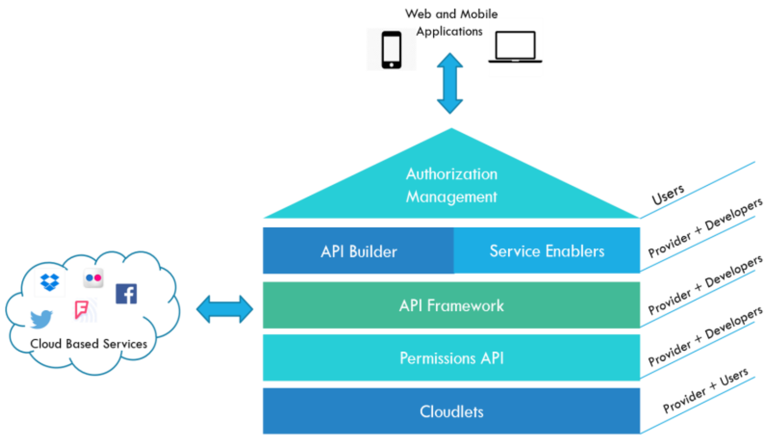


Figure 2. OPENi layering of services and stakeholders

On top of this storage system, a multi-level access control mechanism has been developed, through which users can decide which types of data, or even which data instances, will be accessible by specific OPENi applications. As a privacy-intensive project, OPENi's primary objective is to produce an innovative solution that integrates personal data storage and cloud-based services and gives the end-user maximum control on their data. This requirement is addressed through a number of technical solutions, security and privacy policies: AES and RSA, HTTPS and SSL are deployed to provide transport and storage description; Commercial providers claim different law compliances (mostly US) to their customers, such as HIPAA, PCI, Safe Harbor and ISO27001 compliance as well as different forms of SLAs. While our approach is not to share the focus on US certification, there is a great focus on a compliance with European law. The systems expose a HTTP REST APIs and communicate JSON. OAuth 2.0 is commonly used for access control. However, as oAuth 2.0 is not designed to be a narrow, interoperable standard but rather a framework, the outlined solutions are not interoperable with each other. oAuth 1.0A is supported by some API platforms to access external cloud based services. With the exception of the Intel platform, none of the API solutions target the end-user as a customer. The data belongs to the developer who obtained it. OPENi has a considerably different focus and provides fine-grain access control and detailed audits to end-users to provide them with a user-centric privacy concept.

From the developer perspective, OPENi offers a unified Graph API framework [13] that handles all interactions with the Cloudlets through RESTful services [16], enabling faster and easier application creation. The advantages OPENi has to offer to developers, spanning from its user community, the "Cloudlet" paradigm and the usage of one common unified Graph API to handle all Cloudlet interactions has been discussed in [14]. It is worth mentioning that OPENi also offers its "Builder", a platform addressed to developers, through which they can extend OPENi Objects and APIs, keep versions of their APIs and connect the Objects to other CBS Objects creating mashups, making OPENi sustainable in this rapidly evolving ecosystem [17].

In order to leverage OPENi Cloudlets beyond their storage capabilities, some native OPENi applications, namely the "Service Enablers", have been developed to show the power of this information aggregation in a privacy-aware environment. It is expected that both users and application developers will benefit from the provided Service Enablers, which will function as trusted agents, enabling smarter applications to be built on top of OPENi, following its transparent privacy rules. Four of the Service Enablers (SE) are briefly discussed:

1. The Advertising SE aims at enabling advertisers, under partnerships with service providers, to use OPENi opted-in users' anonymised, personal data in order to enable proficient mobile marketing audience management, as well as targeting optimization and personalization in advertising.
2. The Recommender SE is an extension over the central OPENi architecture that allows developers to build applications enhanced with recommendations natively provided by OPENi, without violating users' cloudlet privacy policy.
3. The Analytics SE provides insights into developers about transactions and usage of their API-enabled applications, relative to OPENi calls and user demographics.
4. The Timeline SE is to provide a common interface for organising user data, enabling the adding, updating, retrieval, and deleting of user data related to time

5. Conclusions

Web and mobile application users are getting progressively accustomed to the idea of having access to their data, regardless of their location or the device they are using. This has caused the dispersion of valuable user information across multiple cloud storage spaces, but also a proliferation of cloud based services that has revolutionized the way these applications address personal and business needs. OPENi brings to both data owners and application developers the Cloudlet, a unified cloud storage space, designed to hold heterogeneous information, leveraging the power of aggregated data in a privacy-aware manner. OPENi makes the data owner and not the service provider in charge of the permission rules enforced on his data. Service providers, through the application developers, can benefit from a more open and trustful relationship with their users, building more powerful applications with the provided unified Graph API. OPENi greatly depends on the idea of communities, both user and developer-wise, so the challenge lays ahead to bring people into this ecosystem. To that end, training platforms will be deployed, courses will be offered and multiple other disseminations actions, including Hackathons, will be held. As another future step, the proposed OPENi ecosystem, will be deployed and tested with other data sources, targeting alternative stakeholders outside of the mobile applications market; thus governments or federal organisations, even enterprises which may use such solutions to improve and make their IT more open and transparent.

Acknowledgments. This work has been created in the context of the EU-funded project OPENi (Open-Source, Web-Based, Framework for Integrating Applications with Social Media Services and Personal Cloudlets), Contract No: FP7-ICT-317883.

References

[1] OPENi Project. OpenSourceProjects repository. https://opensourceprojects.eu/p/openi/

[2] I. Alvertis, T.Kastrinogiannis, R.Kleinfeld, E.Robson et. all, "OPENi Graph API Framework in the Social Standards Landscape," *position paper in W3C workshop on "Social Standards: The Future of Business*, August 7th-8th, 201, San Francisco, USA.

[3] https://www.ctrl-shift.co.uk/research/product/88, accessed online April 2015

[4] www.pewinternet.org/2014/11/12/public-privacy-perceptions/, accessed online April 2015

[5] Innovation Opportunity. Learnings from the midata Innovation Lab https://www.ctrl-shift.co.uk/research/product/81

[6] Datacoup. A personal data marketplace. http://datacoup.com/docs#about

[7] Experian. http://www.experian.com/ , accessed online April 2015

[8] Free is a Lie. Aral Balkan. TNW Europe 2015 presentation. https://www.youtube.com/watch?v=upu0gwGi4FE

[9] Epic Browser. https://www.epicbrowser.com/ , accessed online April 2015

[10] DuckDuckGo. https://duckduckgo.com/, accessed online April 2015

[11] BlueKai. http://www.bluekai.com/ , accessed online April 2015

[12] Smart Disclosuer Policy. https://www.data.gov/consumer/smart-disclosure-policy , accessed online April 2015

[13] Alvertis I., Petychakis M., Lampathaki F., Askounis D., Kastrinogiannis T. " A Community-based, Graph API Framework to Integrate and Orchestrate Cloud-Based Services." AICCSA. 2014.

[14] Petychakis, M., Alvertis I., Biliri E., Tsouroplis R., Lampathaki F., Askounis D. "Enterprise Collaboration Framework for Managing, Advancing and Unifying the Functionality of Multiple Cloud-Based Services with the Help of a Graph API." Collaborative Systems for Smart Networked Environments. Springer Berlin Heidelberg, 2014. 153-160.

[15] Feuerlicht, George, and Hong Thai Tran. "Enterprise Application Management in Cloud Computing Context." Proceedings of the 16th International Conference on Information Integration and Web-based Applications & Services. ACM, 2014. APA

[16] Fielding, Roy Thomas. "Architectural styles and the design of network-based software architectures." 2000.

[17] Tsouroplis R., Petychakis M., Alvertis I., Biliri E., Lampathaki F., Askounis D., "Community-based API Builder to manage APIs and their connections with Cloud-based Services". CAiSE Forum. 2015.

[18] Abbas, Assad, and Samee U. Khan. "A Review on the State-of-the-Art Privacy-Preserving Approaches in the e-Health Clouds." Biomedical and Health Informatics, IEEE Journal of 18.4 (2014): 1431-1441.

[19] Rahimi, M. Reza, et al. "Mobile cloud computing: A survey, state of art and future directions." Mobile Networks and Applications 19.2 (2014): 133-143.

[20] Zhu, Hengshu, et al. "Mobile app recommendations with security and privacy awareness." Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014.

[21] Schwab, K., Marcus, A., Oyola, J. O., Hoffman, W., & Luzi, M. (2011). Personal data: The emergence of a new asset class. World Economic Forum. Retrieved on May 12th, 2015 from http://www3.weforum.org/docs/WEF_ITTC_PersonalDataNewAsset_Report_2011.pdf

Gamifying Software Development Environments Using Cognitive Principles

Position Paper

Naomi Unkelos-Shpigel, Irit Hadar

Information Systems Department, University of Haifa
Carmel Mountain 31905, Haifa, Israel
{naomiu, hadari}@is.haifa.ac.il

Abstract. The topic of enhancing the software development process has received much attention in recent decades. Several models have been developed to this end, typically addressing the characteristics of the process or the organization. We believe that an additional, substantial enhancement of software development can be achieved via encouraging productive behavior among individual software developers. In this paper, we propose a framework for enhancing motivation among software developers, using gamification principles. As a first step in this ongoing research, we designed a prototype for a game, which can be used in various tasks in the software development process. The game is based on cognitive theories, which address motivation among practitioners.

Keywords: Gamification, motivation, flow theory

1 Introduction

Software development processes and their enhancement have been vastly researched in recent years. Many models have been proposed and implemented in industry, replacing the outdated waterfall model, such as the human-centered agile development methodology, and maturity models such as the CMMI.

The use of these models has contributed to the structure and maturity of the software development process. Nonetheless, while progress has been made, many challenges and difficulties still exist, introducing risks to their success. Additional solutions are needed to encourage and guide productive behavior among software engineers to further enhance the software development process and its outcomes [1]. More specifically, we believe that analyzing developers' behaviors and exploring potential solutions from a cognitive perspective would be beneficial to progress this objective.

Several cognitive theories address the topic of motivation in workplaces, including salient examples such as SDT – Self Determination Theory [2], the Flow Theory [3], and the Group Flow Theory [4]. These theories provide guidelines as to how to motivate employees to take active part in the work, and to encourage them to strive for more productive behavior, mainly by encouraging intrinsic and extrinsic motivation [2] and by achieving a state of flow, where the worker is immersed into the task [3].

Persuasive technologies, and specifically gamification, were acknowledged as changing employees' motivation and behavior. Gamification is defined as "the integration of Game Mechanics in non-game environments to increase audience engagement, loyalty and fun" [5, p.2], and was found to encourage users to participate and contribute in computer-supported applications. In recent years, various gamification elements have been embedded in different information systems and applications in general, and, in some rare cases, in applications intended for software engineers in particular. In this research, we aim to explore, and empirically examine, new and effective ways for enhancing software engineering via gamification.

Leveraging on the principles of gamification and based on cognitive motivation theories, our research questions are: (1) What can we learn from cognitive motivation theories toward designing effective gamified environments for software developers? (2) How can we promote software developers' productive behavior via gamifying software engineering practices? (3) What are the actual benefits of embedding gamification techniques in software development environments?

The next section presents the background for our research. Section 3 details our proposed solution. Section 4 presents the planned research method, and section 5 discusses the expected contribution of the research.

2 Scientific Background

2.1 Gamification

Coined by Nick Pelling in 2002 [6] the term "gamification" is used in order to describe how any task can be performed as a game. In recent years, various research works have been conducted with regard to gamification, its mechanisms, and their use. While the early use of gamification was intended for games and application for users, research from the last few years is targeted on using gamification mechanisms for changing behaviors of specific populations for specific purposes. In the context of software development, several attempts to use gamification techniques were conducted.

Sheth et al. [7] gamified a number of software development activities in educational settings, in order to engage software engineering students in development, documentation, bug reporting, and test coverage, using social rewords. The students who used the system showed statistically proven improvement in their work results. The system was later used to encourage students into doing software testing, using a method they called "Secret Ninja Testing," where students were presented with quests using characters from various action movies, and were asked to act as these characters while solving testing problems [8]. The system helped the students to be exposed to the complete lifecycle of software development, and encouraged students to choose software engineering as a major in their studies.

Research was also conducted in the context of using gamification at early stages of software development. Dubois and Tamburrelli [9] suggested a framework to successfully integrate gamification elements into software engineering, starting from requirement elicitation. They identified three types of activities needed to be performed when engaging gamification into software engineering: analysis, integration and evaluation and found that students performing these activities had better results in software engineering. Another attempt to use gamification at early stages of software development showed that using gamification in virtual teams during requirement elicitation assisted the teams to locate experts and share their knowledge [10].Another effort to gamify software development in educational settings was done in the context of early stages of software development, successfully integrating gamification elements into requirement elicitation [11]. This study identified three types of activities needed to be performed when integrating gamification into software development: analysis, integration and evaluation, and found that students performing these activities produced better outcomes.

Thus far, gamification for software engineering has focused on education, using gamification principles borrowed from the domain of applications and website usage. We did not find research in this context relying on cognitive theories in order to design games for software engineers, or using gamified environments in industry in order to motivate practitioners to enhance work performance.

2.2 Motivation Theories

Several cognitive theories address the topic of encouraging motivation for work tasks. Here we briefly present three of the most influential theories in this field.

The Self Determination Theory (SDT) [2] presents a continuum of motivation types, from intrinsic motivation that emerges from the employee, to extrinsic motivation created by rules and regulation in the workplace. Although intrinsic motivation is considered to be linked to positive human behavior, SDT suggests that proper use in extrinsic motivation can lead to motivated behavior. According to the Theory of Flow [3] there are five elements of reaching to a state where the individual is immersed into the performed task (some of which can be extrinsically induced): Clarity, Centering, Choice, Commitment, Challenge. Sawyer [4] extended these elements to the context of group flow, to contain, among others, the following characteristics: A compelling, shared goal, a sense of being in control, blending egos, equal participation, familiarity, constant and spontaneous communication, and the potential for failure. We relied on these characteristics when designing our solution, creating an environment that would encourage group flow. The proposed solution is described in the next section.

3 The Proposed Solution: Gamifying Software Development

The objective of this ongoing research is to identify different opportunities within the software development process for enhancing productive behavior via gamification. In this paper we refer, as examples for demonstrating our vision, to six of the major tasks of the software development process. We view them in pairs:

- Software architecture design and software architecture review
- Coding and code review
- Customization (adapting the solution to the customer) and Integration Testing

We chose to focus on these pairs since we can identify in each pair two parties: The creator (architect, programmer, customizer) and the reviewer (architecture reviewer, code reviewer, tester). Each of these pairs will have its own game, according to the following principles (see Fig. 1):

Create – The creator creates a segment of work, according to her task (architecture, coding or customizing). Creating the segment assigns points to the creator and to her team.

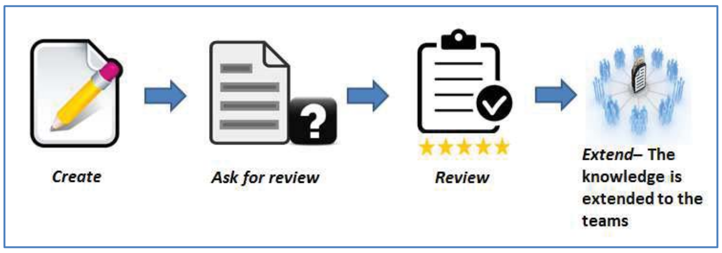
Request review – The creator sends the artifact for reviewer. The reviewer is randomly selected from a group of potential reviewers. The creator does not know which reviewer was selected. The reviewer and the team receive points for this action.

Review – The reviewer receives the anonymous artifact, reviews it and writes a review. She then submits the review to the system, with a review score, which reflects the quality of the artifact. The reviewer and the team receive points for this action. Additional points are given to the reviewer for writing comments for improvement.

Extend the knowledge – upon receiving the review, the creator and reviewer can further extend the knowledge created from the review (the artifact and the review comments), to their team or to an extended group of practitioners in the organization. Each such knowledge extension results in additional game points for the creator, the reviewer and the team. When the knowledge is used (actively checked-out) by another practitioner from the organization, the team is given additional points.

Fig.1 describes the principles of the game:

Figure 1. The principle of the game of CARE



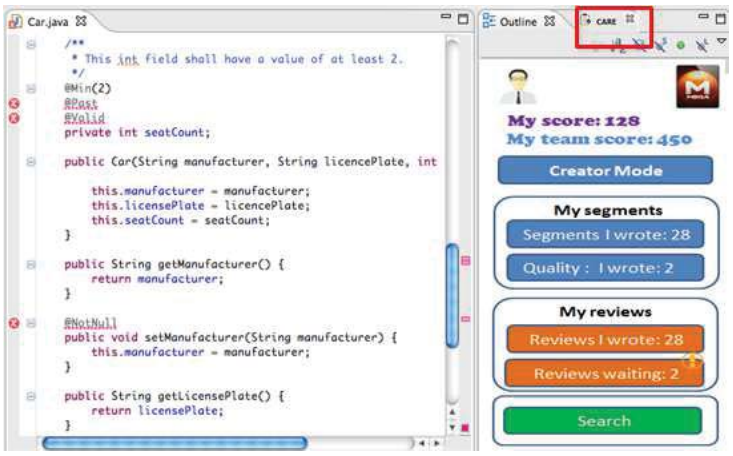
The game we designed has several key principles:

1. The game is embedded in the eclipse IDE. There is a special tab for the game in IDE, where the players can view their profile and all game related information.
2. Each player can see her profile as a creator or as a reviewer (see *Creator/ Reviewer Mode* button in Fig. 2 and 3 respectively). The creator's screen contains information about previous segments created and their average quality score (see Fig. 2, *My segments*). The reviewer's scree contains a section, which refers to the current segment (Fig. 3, segment of code marked in red). She can write a textual review, and provide a quality score for the segment (Fig. 3, *Segment review*).
3. When a segment of code is created and ready, the creator asks for a review, and is immediately rewarded with points.
4. The reviewer reviews the relevant segment, inserting both comments and a quality score. If the reviewer approves the code, she is granted with points as well. Additional score is given for writing a review, which helps the programmer to

improve the code. For bug detection, the reviewer will be rewarded extra points for each bug found.

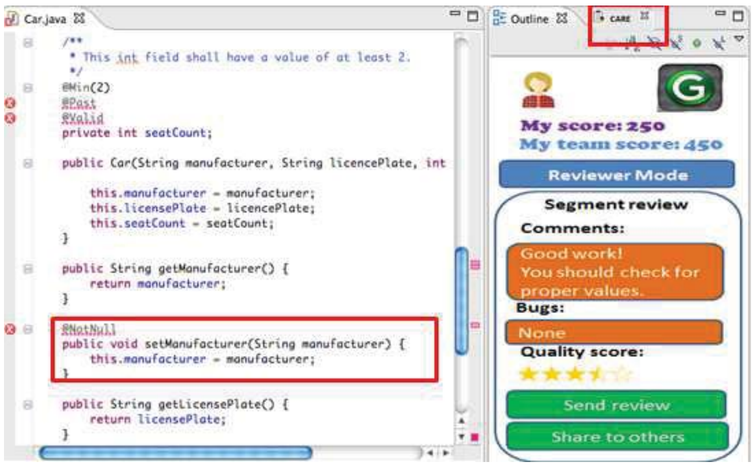
5. The reviewers can choose to share their review comments with members of other teams (pending creators' permission), raising both individual and team score. When the reviewer wishes to share the comment, a message will be prompt to the creator to ensure he agrees to share the segment and the comments. The names of the teams that used the comments are displayed in the team`s profile (see Fig.4, bottom). Additional mechanism is needed to evaluate the quality of the shared information, and its contribution to other stakeholders in the project.
6. The creator can also search for tips and lessons learnt from the review with other creator and reviewers (Fig. 2, *Search*). Using this knowledge (by checking it into the project) raises both individual and team score
7. The creators and reviewers are also given badges according to their individual scores. The badge indicates their level in the game, labeled: kilo, mega, or giga, etc., according to the number of points they earned (Fig 2 and 3, top). All the badges of the team members are displayed in the team`s profile, sorted in groups according to levels (Fig. 4).
8. In addition to the individual scores, there is also a team score managed, which is updated according to the individually rewarded tasks (Fig. 2, top).
9. The teams are rewarded each month according to their scores. The reward can be in the form of monetary incentive or other rewards (e.g., breakfast with a high management representative or coupons for fun activities).
10. If other creators or reviewers use the knowledge and tips shared, the individual who wrote and/or shared this knowledge receives additional points.

Figure 2 .The creator`s screen



```
/**
 * This int field shall have a value of at least 2.
 */
@Min(2)
@Past
@Valid
private int seatCount;

public Car(String manufacturer, String licencePlate, int

    this.manufacturer = manufacturer;
    this.licensePlate = licencePlate;
    this.seatCount = seatCount;
}

public String getManufacturer() {
    return manufacturer;
}

@NotNull
public void setManufacturer(String manufacturer) {
    this.manufacturer = manufacturer;
}

public String getLicensePlate() {
    return licensePlate;
}
```

Figure 3 .The reviewer`s screen

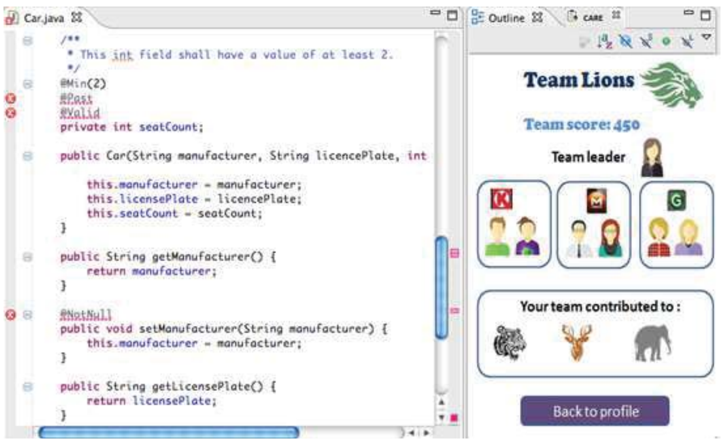


```
/**
 * This int field shall have a value of at least 2.
 */
@Min(2)
@Past
@Valid
private int seatCount;

public Car(String manufacturer, String licencePlate, int

    this.manufacturer = manufacturer;
    this.licensePlate = licencePlate;
    this.seatCount = seatCount;
}

public String getManufacturer() {
    return manufacturer;
}

@NotNull
public void setManufacturer(String manufacturer) {
    this.manufacturer = manufacturer;
}

public String getLicensePlate() {
    return licensePlate;
}
```

My score: 250
My team score: 450

Segment review
Comments:
Good work!
You should check for proper values.
Bugs:
None
Quality score:

Figure 4 .The team's screen



```
/**
 * This int field shall have a value of at least 2.
 */
@Min(2)
@Past
@Valid
private int seatCount;

public Car(String manufacturer, String licencePlate, int

    this.manufacturer = manufacturer;
    this.licensePlate = licencePlate;
    this.seatCount = seatCount;
}

public String getManufacturer() {
    return manufacturer;
}

@NotNull
public void setManufacturer(String manufacturer) {
    this.manufacturer = manufacturer;
}

public String getLicensePlate() {
    return licensePlate;
}
```

Team Lions
Team score: 450
Team leader
Your team contributed to :

The game includes the following gamification elements:

Personal profile – the team members have individual profiles, where they can view their personal and team score. Each team has a team profile, presenting the team members and the team's score.

Badges – the team members are assigned with badges according to their individual scores. The badge indicates their level in the game, according to the number of points they earned.

Scoreboard – each team has its members rated by their scores, and at the end of each month one team member is rewarded as "the player of the month."

The game supports SDT[2], since it offers rules and regulation (in the form of a game), to encourage employees to ask for review, and to share their knowledge with practitioners outside of their team, thus creating extrinsic motivation.

According to the theory of flow, the conceptual design of the gamified environment we propose supports the five elements of flow [3]: *Clarity* – The game and scoring in the game are simple and clear; *Centering* – the game is designed to make players feel they are in the center, gaining individual points and making their contributing to the team visible; *Choice* – players can choose whether to share their knowledge; *Commitment* – since all the players are team members, they are encourages to perform activities which raise team score; *Challenge* – the game provides challenges to all the stakeholders in the process, when they are required to improve the quality of their work, or the work they review, in order to earn additional individual and team points.
The game`s conceptual design also supports the group flow elements [4]: *A compelling, shared goal* – all the players have the shared goal of getting a high team score; *a sense of being in control* – since players send their work when they choose, they have full control on their progress in the game; *blending egos* – since there is a team goal, along with the personal goal, all the players' egos are blended to achieve high team score; *equal participation* – each of the players is allowed to participate in the game equally; *familiarity* – all the players in the same team are personally familiar (with at least part) with each other; *constant, spontaneous communication* – the virtual game allows all the players to communicate with each other; *the potential for failure* –As there is an ongoing race among the individual players and among the teams, low achievements relative to other players can be considered as failures..
To conclude, our proposed gamified environment supports principle of three cognitive theories – referring to individual and group motivation – designed to meet the challenge of achieving full commitment to the task, from both individual and team points of view.

4 Validation

This ongoing study will apply both qualitative and quantitative research method for validating and further refining CARE. As derived from the research objective and questions, the research focuses on human-related processes, calling for the use of qualitative research methods [12], and on performance, which can be quantitatively measured.

The qualitative study will focus on understanding how different gamification techniques affect software developers' motivation and behavior. The main population of this study will be software developers with different levels of seniority. Additionally, as the research settings and its preliminary results will require, we will expand the research population to other roles within the software development process.

Following the findings of the qualitative study, we will refine our conceptual design of the gamified development environment. This design will be implemented and evaluated according to the principles of design research [13]. More specifically, we will focus on measuring the quality and quantity of the software developed, with and without the use of the gamified environment. We plan to conduct this study with the participation of software engineering students and, if possible, practitioners, to find if, and to what extent these means improve their performance and promote desired behavior.

5 Expected Contribution

Gamification has been quite thoroughly researched among different types of users in recent years. However, we find only few examples of gamification research in the context of software development, most of which are intended for students and discussed in the context of education.

Since we wish to contribute to the software and information systems engineering industry, we plan to elicit data and validate our findings and results with practitioners, thus receiving non-biased opinions, aiming at the target population. The results of our study are expected to help organizations in increasing software developers' motivation to complete their tasks successfully and efficiently.

This research will contribute to the academic research community by providing empirical insights into the use of gamification as a means for enhancing software development processes, and the cognitive and social implications thereof.

6 References

[1] Hadar, I.: When Intuition and Logic Clash: The Case of the Object Oriented Paradigm, Science of Computer Programming, vol. 78 (2013), 1407-1426 (2013)

[2] Ryan, R. M., and Deci, E. L.: Self-determination theory and the facilitation of intrinsic motivation,social development,and well-being. American psychologist, 55(1),68 (2000)

[3] Csikszentmihalyi, M.: Flow and the Psychology of Discovery and Invention. Harper Perennial, New York (1997)

[4] Sawyer, K.: Group Genius: The Creative Power of Collaboration, Basic books.(2008)

[5] Deterding, S., Khaled, R., Nacke, L., and Dixon, D.: Gamification: Toward a definition. In CHI 2011 gamification Workshop Proceedings ,12-15 (2011)

[6] Hägglund, P.: Taking gamification to the next level (2012)

[7] Sheth, S. K., Bell, J. S., and Kaiser, G. E.: Increasing Student Engagement in Software Engineering with gamification (2012)

[8] Bell, J., Sheth, S., and Kaiser, G.: Secret ninja testing with HALO software engineering. Proceedings of the 4th int'l worksop on Social software engineering ,43-47ACM (2011)

[9] Dubois, D. J., and Tamburrelli, G.: Understanding gamification mechanisms for software development. In Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering , 659-662 ACM (2013)

[10] Marshburn, David G. and Henry, Raymond M.: Improving Knowledge Coordination in Early Stages of Software Development. SAIS 2013Proceedings. Paper 23 (2013)

[11] Hevner, A. R., and March, S. T.: The information systems research cycle. Computer, 36(11), 111-113 (2003)

[12] Bogdan, R. C., and Sari K. B..: Qualitative research in education. An introduction to theory and methods. Allyn & Bacon, A Viacom Company, 160 Gould St., Needham Heights, MA 02194; (1998).

[13] Hevner, A. R., and March, S. T.: The information systems research cycle. Computer, 36(11), 111-113 (2003).

Community-based API Builder to manage APIs and their connections with Cloud-based Services

Romanos Tsouroplis[1], Michael Petychakis[1], Iosif Alvertis[1], Evmorfia Biliri[1], Fenareti Lampathaki[1], Dimitris Askounis[1]

[1] National Technical University of Athens, School of Electrical and Computer Engineering, Athens, Greece
{ rtsouroplis, mpetyx, alvertisjo, ebiliri, flamp, askous} @epu.ntua.gr

Abstract. Creating added value, innovative applications and services in the web is hindered by the prevailing different formats and models of information retrieved by the various Cloud-based Services (CBS). Additionally, CBS tend to change their Application Programming Interface (API) versions very often, not sufficiently helping the interested enterprises with their adoption as they need to be up-to-date and always functional. The API Builder is a community-based platform that aims to facilitate developers in adopting a Graph API that unifies the experience of multiple cloud-based services APIs and personal cloudlets, building and maintaining their software applications easily, despite any changes made in the CBS APIs.

Keywords: APIs, Cloud-based Services, Evolving APIs, Graph API, Community-based platform, Social Enterprise

1. Introduction

These days, more and more users tend to share their data through the World Wide Web. Social Networks have gathered large silos of information for each of their users and their interconnections. The Web APIs (Application Programming Interface) provide all the endpoints needed for a developer to request access into this plethora of information. All the major social networks like Facebook, Twitter, etc., have an API, providing most of their core services to third parties for several reasons [1].

Currently, on the site Programmable Web, which indexes, categorizes and makes mashups of APIs, 12,987 public APIs are listed and this number is continuously growing [2]. This, along with the fact that there is not a standardized, specific model representing each object in the world make the differentiations among the API objects a huge overhead for anyone that needs to integrate a product with various Cloud-based Services. In fact, there are so many services offering APIs for their nodes that a developer or even a team of developers within a company would have difficulties keeping up with the always evolving API landscape.

Companies, developers and end-users are directly affected by the great changes that CBS APIs go through continuously. In particular, Facebook API, one of the most used, is now on version 2.2 and all the client application using version 1.x have until

April 30, 2015 to upgrade to v2.0 or later, as they are going to be deprecated and not functional afterwards. Other APIs are developed keeping in mind that future versions will come, ensuring the end-users that what exists now will remain functional, like Dropbox claims to. Last but not least, many APIs become discontinued, like Desktop API by Skype. To sum up, APIs are born, evolve and die just like living organisms and there is need of great effort to keep up with them [3].

The effects that the Cloud-based Services API changes create on the developer community have been thoroughly researched [4, 5]. This is the main challenge we try to confront, so as to offer enterprises/developers the ease of mind that is needed for considering usage of one or multiple APIs when building their software products and services. The OPENi API Builder uses the power of community over a clean and simple user interface to produce an updated Generic API recommendation, based on valid REST API standards, matching the most used objects of a great number of Cloud-based Services APIs.

The structure of the present paper is as follows. Section 2 presents the methodology followed towards the implementation of the OPENi API Builder. In section 3, an overview of the API Builder is given, describing the added value features provided, broken up into 3 smaller sections, the community-based orientation, the documentation in all the standards currently available and lastly the case of CBS API changes handling. In section 4 we provide related work and compare it with the Builder. Section 5 concludes the paper with future work on this matter.

2. Methodology

Due to the high complexity of managing and orchestrating the changes in all of the CBS APIs, a mechanism that would handle these changes had to be implemented. But first, a research was conducted on the field of API creation. This was completed in 7 stages, based of course on the modeling already conducted for the Graph API [6, 7]. Validation of each step and peer-reviewing of the work done helped throughout the whole process to keep constantly up-to-date.

- **Step 1. Identifying the needs for handling the Graph API Evolution.** In this step, user stories were created to describe the expected functionalities for future updates on the Graph API Platform APIs, Objects, methods and CBS.
- **Step 2. Research of various similar solutions.** API creation tools can be found in many enterprise solutions across the Web, on studies and even as open source framework for specific code languages. All of these solutions were gathered and categorized based on the providing features.
- **Step 3. Connectivity with CBS.** The CBSs add extra value to a business solution as already described. Categorization on the CBS has already taken place for the Graph API Platform [6].
- **Step 4. Automation of the evolution process.** Research [3] has been conducted on the automation of the APIs evolution. There is no solution that does not involve examining the CBS backend code, so a semi-automatic approach was chosen based on a swift-responding to changes community.

- **Step 5. Building upon standards.** All of the available solutions depend on one or none of the thriving standards on API documentation. Support on the most significant of them was decided after carefully exploring them and their impact on the developer community.
- **Step 6. Identifying scalability concerns.** The Builder should be able to manage multiple simultaneous user actions and keep a huge amount of data for objects and interconnections with CBS, so an analysis of best practices towards this end was taken into consideration.
- **Step 7. Crafting UI mockups** to better express the power of the Builder. After assembling the requirements for making the API Builder, basic interface design principles like intuitiveness, familiarity, simplicity, availability and responsiveness were followed in order to ensure a great user experience.

3. Overview

The OPENi API Builder is a Web-based platform, implemented as an open source project, publicly available in Github [8].

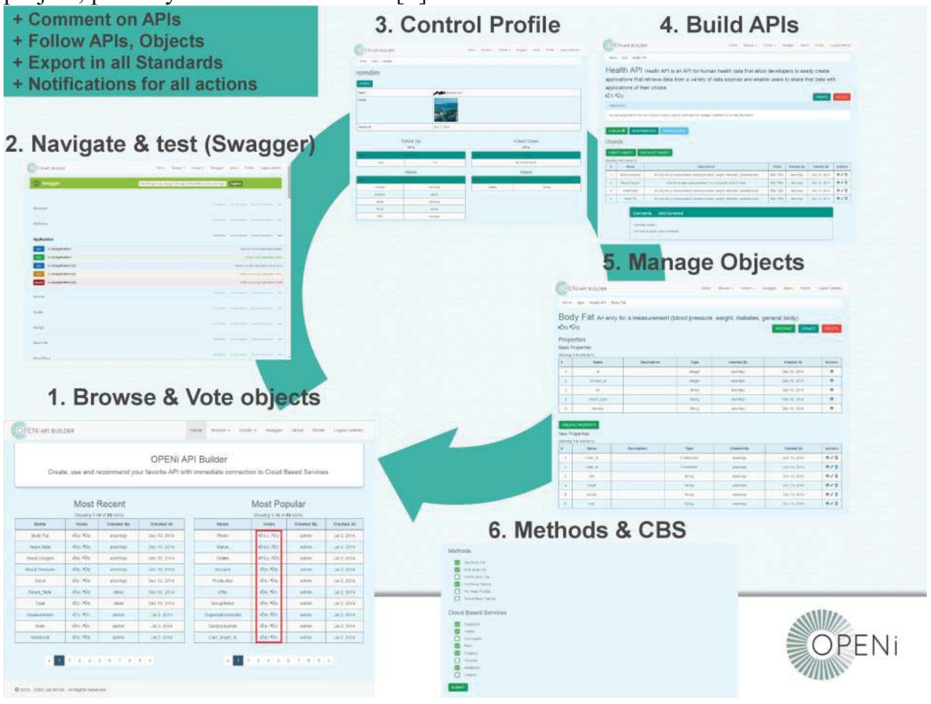


Figure 1 - API Builder Actions

Through its web interface, developers can create, update, delete their own APIs, duplicate existing ones, manage CBS API instances and create correspondences between them. In order to allow for more flexibility, APIs inside Builder consist of multiple components which can also be modified independently. More specifically, they have the following structure: APIs can have multiple Objects and Objects connect with Properties, Methods and CBS.

An authenticated developer can browse public APIs and Objects, vote them up/down, follow them or their creators as shown in Figure 1. API specification and testing is provided through a swagger interface embedded in the website. The provided level of automation in API handling helps developers focus on more meaningful tasks and deliver their work more quickly, increasing business profits.

The innovation of the API Builder can be found in 3 different features, the community backing the API and CBS management along with the social characteristics that are provided, the standards on which the Builder structures its documentation and the Cloud-based Service APIs evolution handling.

3.1 Community Orientation

The transformation of the World Wide Web these last years, from a static directory of connected but one-person authored files to the Social Web as we know it today, usually referenced as Web 2.0, showed the dynamics of the crowd in crafting truly valuable information. From the Linux Kernel [9] to the Microsoft .NET tools and frameworks, the open source solution gives the opportunity to build a large community around business products that make the end-users interact and bind with them. Many papers have been written for the advantages of transforming a traditional firm-based model into a community-based development process [10, 11].

The OPENi API Builder has been built to accommodate all the needs of an active community. In Table 1 we can see the actions that are allowed for an authenticated enterprise user. The Builder has a lot of social characteristics as well, allowing the users to interact and engage with each other as well as with their products. The influence indicator of an API for example can be calculated by taking into account the votes and the number of users following that particular API.

Builder	User Actions							Social Characteristics				
Parts	Create	Duplicate	View	Update	Delete	Select	Publish	Propose	Vote Up	Vote Down	Comment/Reply	Follow
API	Y		Y	Y	Y		Y	Y	Y	Y	Y	Y
Object	Y	Y	Y	Y	Y				Y	Y		Y
Property	Y		Y	Y	Y							
Method						Y						
CBS	Y		Y	Y		Y		Y	Y	Y		
Other Users			Y									Y

Table 1 - User Actions & Social Characteristics

In order to avoid duplicate declarations of Objects and promote reusability of objects inside the community, aiming at more sustainable APIs, during the creation of a new object the developer is presented with recommendations of existing ones that could be used instead, based on a combination of approximate string matching on the selected names and similarity computation of the types of objects.

3.2 Documentation based on standards

In order to provide a consistent way to document APIs, a number of REST metadata formats has been created. These standards offer a way to represent an API by specifying entry point(s), resource paths, methods to access these resources, parameters to be supplied with these methods, formats of inbound/outbound representations, status codes,error messages and documentary information.

Some of the most popular standards are: Swagger, which offers a large ecosystem of API tooling, has very large community of active users and great support in almost every modern programming language and deployment environment. API Blueprints, where an API description can be used in the apiary platform to create automated mock servers, validators etc. The Hydra specification [12], which is currently under heavy development, tries to enrich current web APIs with tools and techniques from the semantic web area. RAML is a well-structured and modern API modelling language. And finally WADL, an XML-based specification submitted to W3C, but not widely adopted. Table 2 show more information about these standards. Swagger is obviously the dominant choice for the moment, though all specs are promising.

In order to allow developers to interact with a wider range of APIs independently of their preferred standard, an API can be exposed through the Builder to any of the aforementioned specifications. Moreover, a format transformation component has been implemented, enabling transformation amongst these five specifications.

Details\Specs	API Blueprints	RAML	Hydra	WADL	Swagger
Format	Markdown	YAML	Hydra Core Voc. - JSON-LD	XML	JSON
Licence	MIT	ASL 2.0/TM	CC Atrribution 4.0	Sun Microsystems Copyright	ASL 2.0
Available	Github	Github	www.w3.org	www.w3.org	Github
Sponsored By	Apiary	Mulesoft	W3C Com Group	Sun Microsystems	Reverb
Version	Format 1A revision 7	1.0	Unofficial Draft 19 January 2015	31 August 2009	2.0
Initial Commit	April 2013	Sep 2013	N/A	Nov 2006	July 2011
Pricing Plans	Y	Y	N	N	N
StackOverflow Questions	75	37	35	156	732
Github Stars	1819	1058	N/A	N/A	2459
Google Search	1.16M	457k	86k	94.1k	28M

Table 2 – API Specification Details

3.3 CBS API changes handling

When a Cloud-based Service decides it is time to change its API, it may deliver additions, deletions and/or alterations on existing Objects and/or the url schema itself. This is a huge overhead for enterprises as specified in the introductory section. It can lead to great increase of the development cost to cover new CBS documentation and code refactoring. The OPENi API Builder provides, to the best of our knowledge, the easiest way to adapt all these changes through the swift reflexes of its community, hence saving time and expenses from enterprises.

All the Objects created at the Builder can have their properties mapped to the corresponding properties of a similar Object of some CBS. These mappings are essentially arrays of labels, where each label represents a property from this particular CBS API. Following version changes is as easy as drag and dropping new labels, removing the deprecated ones and renaming those that need to be altered. Extra attention needs to be given though to the required properties of an Object and the methods that can be applied to each Object. Through a clean user interface, the process of making a match with an API is easier than using wrapper code in some programming language. Even the url that is needed to make the calls can be provided and altered in plain text.

When a new matching version is ready, developers can propose this API for implementation in the official OPENi Platform, and create the documentation in all available specifications. Through the Builder social characteristics, the community advances the most used and up-to-date APIs, promoting them for enterprise solutions.

4. Related Work

Several alternative business and developer-oriented solutions exist for creating and managing APIs, based on various or no standards. StrongLoop Arc provides a nice user interface for implementing APIs that are then rendered via a Swagger-based API explorer. Alas, there is no community built upon this solution. Apiary uses API Blueprints code to provide the documentation along with additional features. The downside is that the API management is accomplished via the code. The Apiary solution has a community but no social characteristics. API Designer by Mulesoft has no visual interface as well. The developer writes RAML code to create an API. Appcelerator, a company which targets mobile apps, gives an interface for rapid assembly and hosting of mobile-optimized APIs with some prebuilt connectors. Apigility, an open source tool created by Zend Framework, running on own hosting environment, has visual interface for creating APIs with the ability to produce documentation in Swagger. Apigee Edge API Platform innovates in providing good analytic services for the APIs while providing most of the available CBS connectors. And last but not least, Kimono, by kimonolabs is a very interesting tool which allows users to scrape a website, by simply selecting data on the site. The tool then gathers similar information from the site into an endpoint for a new API. It does not have the same flexibility in managing your API but instead tries to automate the process in existing websites. No API documentation standard is yet provided. All of the feature variations can be found in Table 3 below.

Almost all of these solutions lack the social characteristics that the Builder provides. The most important differences though are the support of multiple available standards for documentation and the mapping between Cloud-based Services and the created APIs that keep track of all the changes in versions.

Features\Solutions		StrongLoop Arc	Apiary	API Designer	Appcelerator	Apigility	Apigee	kimono	API Builder
	user interface	Y			Y	Y	Y	Y	Y
	open source					Y			Y
	community		team-oriented	Y	team-oriented			Y	Y
	social network		Y	Y				Y	Y
Specification	Swagger	Y				Y	tweaked ui		Y
Specification	Hydra								Y
Specification	RAML			Y					Y
Specification	WADL								Y
Specification	API Blueprints		Y						Y
	Datastore Connectors	Y					Y		Y
	Connectivity with CBS				static		Y		dynamic
	Matching with CBS								Y

Table 3 - Related Work

In this context, the User Interface is the visual representation of the APIs management board through buttons, textboxes and labels. When no UI is provided, coding is used for APIs administration. The only open source alternative to the API Builder is Apigility. All the other Platforms are products developed by companies and have various pricing plans. Most of them have community features, though in Apiary and Appcelerator it has the form of a team-based product with cooperation features varying according to the pricing plans. Apiary, API Designer and kimono have social features like commenting, following and liking other projects. For Apiary this can be seen only inside the team collaborating on a project. Extra attention has been given to the OPENi API Builder on this matter as already described before in section 4.1.

For the documentation standards supported, StrongLoop Arc and Apigility follow Swagger, while Apiary is based on Api Blueprints and API Designer on RAML. Apigee is among the first ones supporting Swagger. To the best of our knowledge, only the API Builder supports all forms of standards for the documentation. Datastore connectors can be all sorts of SQL and NoSQL databases, like MySQL, MongoDB, SQL Server, etc. StrongLoop Arc and Apigee support integration and retrieval of these databases' models for creating APIs as well as the API Builder.

Connectivity with CBS is predefined for Appcelerator for a range of enterprise and public data sources like SAP, Oracle, Facebook, Twitter, etc. In the API Builder the Cloud-based Services can be altered dynamically through the community. Last but not least, the feature of matching Objects of APIs to other Objects from CBS can only be met at the API Builder as described before in section 4.3.

5. Conclusion

OPENi API Builder delivers a unique proposition for enterprises that want to go the extra mile in getting ahead of their competition. Utilizing the power of the community for seamless integration with multiple Cloud-based Services, the API Builder has a great value as a platform for enterprises, start-ups and freelancers. There is no need to keep track of the latest changes in CBS APIs, reassuring continuous uptime in own services. Easy to use and clear, Builder delivers documentation for its APIs in the most popular standards, providing compatibility with most business needs. OPENi API Builder, as a work in progress has to be further expanded and tested to fully discover its potentials and extend it towards the community's needs. The code is open source, GNU licenced and can be found at Github repository.

As future work we intend to disseminate the power of the Builder and increase its usage among developers/companies. We are also exploring the automated management of Cloud-based Services through different versioning, by parsing a documentation in one of the standards to automatically create the whole API in the Builder. This has already been implemented for Swagger documentation for the needs of interconnection between the Graph API Platform and the API Builder. Since the major documentation standards can be transformed from one to another as described in section 4.2, what remains is to determine the versioning and deprecation policy required to guarantee the most comforting solution for developers, start-ups and enterprises when using the API Builder.

Acknowledgments. This work has been created in the context of the EU-funded project OPENi (Open-Source, Web-Based, Framework for Integrating Applications with Social Media Services and Personal Cloudlets), Contract No: FP7-ICT-317883.

References

1. A Survey of the API Economy. Israel Gat, Giancarlo Succi. Agile Product & Project Management. Executive Update Vol. 14, No. 6
2. Programmable Web API Directory. http://www.programmableweb.com/apis. [Accessed: 28-02-2015]
3. Espinha, Tiago, Andy Zaidman, and Hans-Gerhard Gross. "Web api growing pains: Stories from client developers and their code." *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on*. IEEE, 2014.
4. Wang, Shaohua, Iman Keivanloo, and Ying Zou. "How Do Developers React to RESTful API Evolution?." *Service-Oriented Computing*. Springer Berlin Heidelberg, 2014. 245-259.
5. Robbes, Romain, Mircea Lungu, and David Röthlisberger. "How do developers react to api deprecation?: the case of a smalltalk ecosystem." *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*. ACM, 2012.
6. Alvertis I., Petychakis M., Lampathaki F., Askounis D., Kastrinogiannis T. " A Community-based, Graph API Framework to Integrate and Orchestrate Cloud-Based Services." AICCSA. 2014
7. Petychakis, M., Alvertis, I., Biliri, E., Tsouroplis, R., Lampathaki, F., Askounis D. "Enterprise Collaboration Framework for Managing, Advancing and Unifying the Functionality of Multiple Cloud-Based Services with the Help of a Graph API." *Collaborative Systems for Smart Networked Environments* (2014): 153-160.
8. OPENi API Builder source code at Github, https://github.com/OPENi-ict/api-builder
9. Lee, Gwendolyn K., and Robert E. Cole. "From a firm-based to a community-based model of knowledge creation: The case of the Linux kernel development." *Organization science* 14.6 (2003): 633-649.
10. Shah, Sonali K. "Community-based innovation and product development: findings from open source software and consumer sporting goods." (2003).
11. Krishnamurthy, Sandeep. "An analysis of open source business models." (2005).
12. Lanthaler, Markus, and Christian Gütl. "Hydra: A Vocabulary for Hypermedia-Driven Web APIs." LDOW. 2013. APA

Applying Idea Management System (IMS) approach to design and implement a collaborative environment in public service related open innovation processes

Marco Alessi[1], Alessio Camilo[1]✉, Valentina Chetta[1], Enza Giangreco[1],Mona Soufivand[1]✉, Davide Storelli[1]

[1]*Engineering ingegneria informatica SPA, R&D Department, Lecce, Italy*
alessio.camillo@eng.it, valentina.chetta@eng.it, enza.giangreco@eng.it, davide.storelli@eng.it, marco.alessi@eng.it, m.soufivand@gmail.com✉

Abstract. Novel Ideas are the key ingredients for innovation processes, and Idea Management System (IMS) plays a prominent role in managing captured ideas from external stakeholders and internal actors within an Open Innovation process. Considering a specific case study, Lecce-Italy, we have designed and implemented a collaborative environment which provides an ideal platform for government, citizens, etc. to share their ideas and co-create the value of innovative public services in Lecce. The application of IMS in this study with six main steps, including: idea generation, idea improvement, idea selection, refinement, idea implementation and monitoring shows that this, remarkably, helps service providers to exploit the intellectual capital and initiatives of the regional stakeholders and citizens, and assist service providers to stay in line with the needs of society.

1 Introduction

Value co-creation is one approach to create innovative services. Co-creation is the process by which products, services and experiences are developed jointly by companies, their stakeholders and final customers, opening up a whole new world of value [19]. It is a new way of thinking about providing public services in a reciprocal relationship between service providers, professionals, service users and citizens, which makes such services much more effective, efficient, and far more sustainable [4]. Progress in technologies such as Web 2.0 phenomenon [18], offers the ideal platform for service providers, users and other actors to communicate and interact with each other for exchanging ideas and opinions, which are necessary (but not sufficient) to foster the process of value co-creation. Great ideas are the key parameters of innovation process for organizations and communities. The ideas flowing without a proper managing mechanism to evaluate, categorize and prioritize them, would not assist innovation process. As stated by Geoff Mulgan [17], ‘’Innovation is often given complex definitions’’ but he prefers the simple one: ‘‘new ideas that work’’. Reviewing related literature shows the importance of ideas in the innovation processes. As an example, the European Foundation for Quality

Management (EFQM)[1] defines innovation as "the practical translation of ideas into new or improved products, services, processes, systems or social interactions". Ven and Poole (1989) [24] argue that "invention is the creation of a new idea, but innovation is more encompassing and includes the process of developing and implementing a new idea. The development of innovation is not a linear process (a pipeline of sequential processes), but it needs a systemic approach". Therefore, Innovation starts with 'management of ideas' [3]. The formal process such as Idea Management System (IMS) to structure the aforementioned stages including: capture, filter, evaluation and implementation of the best ideas, seems essential. Lack of this system may cause superfluous innovation efforts [23]. The complex interactions between many individuals, organizations and their operating environment is an open innovation process [5,][6]. Chesbrough defines open innovation as: "the use of purposive inflows and outflows of knowledge to accelerate internal innovation, and expand the markets for external use of innovation, respectively. Open innovation is a paradigm assuming that firms can and should use external ideas as well as internal ideas, and internal and external paths to market, as the firms look to advance their technology" [7] [6] [21]. The Open Government (OG) concept, which emphasizes on including citizens and society as well as administration members within governmental processes, is a translation of open innovation in governmental processes. OG seeks to engage citizens in order to increase efficiency within political/organizational decision process leading to society's satisfaction [10].

2 Problem Description

According to the Edelmann, governments are aware of the significance of citizens engagement in decision making processes by integrating their potential in innovation process and acquiring better outcome [10] which reflects a paradigm shift in public administration; however, as stated by Collm & Schedler, innovation process in public sector up to now has occurred in closed-off processes mainly handled by internal public administration and sometimes with the consultancies support [8].

Public administration has understood the need to encourage stakeholders and citizens to participate nevertheless, it still has not found its role in the virtual atmosphere [10]. The ubiquitous presence of ICT together with the recent willingness of citizens to participate and contribute online, can enable government agencies to restructure their interaction with citizens in order to achieve better collaboration results [13]. IMS has been successfully implemented in private sector with the purpose of identifying the real demands in order to generate services and products based on them [23]. However, the current discussion on open innovation has hardly touched upon the public sector. For example, Brunswicker has investigated the possibility to applicate crowdsourcing platform in the governmental context. These studies showed that design principles derived from open innovation projects in the corporate world may not be directly applied in the governmental context: they need to be adjusted and integrated [15]. In this study, we propose a conceptual framework describing the idea of life-cycle and the tools enabling collaboration between citizens

[1] http://www.efqm.org/

and Public Administration, with particular focus on Idea Management System and its role in each step. In this paper, we present the Idea Management System[2] developed in the Puglia@service[3] project supporting the co-creation activities in the initiative for Lecce candidacy as European Capital of Culture 2019[4].

2.1 Case Study

The Municipality of Lecce (Italy) has decided to change the approach of creating a shared path towards a social model in which a direct participation and collaboration of the citizens is included in order to generate innovation.

Public administration and citizens are generally not coordinated with each other, since the traditional approach of urban planning is top-down and often does not match citizen needs. Citizen's involvement and their needs definition are important elements for Lecce. For these reasons the Municipality of Lecce organized LUAC`s (urban, open, creative laboratories) which are a kind of informal debate aiming to satisfy citizens' participation. "Lecce 2019 – Idea Management System" was adopted to integrate LUAC`s and other initiatives that enable interaction between citizens.

The implementation of the "Lecce 2019 - IMS" was performed using the tool Gi2MO IdeaStream[5]. It consists of a set of modules able to customize Drupal [9] in order to implement it as a system of Idea Management.
As for access to the platform, the correlation between the number of visits and the interest shown by citizens and local associations towards the initiative Lecce in 2019 is evident. The launch of the website, which took place in July 2013, was accompanied by a steady increase with a peak in September, close to the deadline for submission of the bid book[6]. From then on there was a decline in the month of November at the announcement of the results of the first phase of selection, which shows that the number of accesses and interactions is strongly influenced by the diffusion of the various initiatives and different maturities (See Table 1).
In this regard, Caritas Diocesana of Lecce proposed, within the IMS, the idea of creating a network of solidarity aimed at collecting and distributing food. This idea has been voted and commented by other voluntary associations (Red Cross of Lecce, Comunità Emmanuel, etc.) and by some local shops, all enthusiastic and ready to participate. The Municipality of Lecce by considering the idea interesting for the local community and evaluating, thanks to sentiment analysis indicators, the interest shown on the web for this topic, has intervened, proposing itself as guarantor and coordinator of this network. Meetings and focus groups were organised in order to create a "network of solidarity" involving several actors, such as: voluntary associations, Confcommercio, Confesercenti, Confindustria and the managers of the Puglia@service project. The latter have given their availability to implement a

[2] http://www.2019idee.eu/
3 http://www.pugliasmartlab.it
4 http://ec.europa.eu/culture/tools/actions/capitals-culture_en.htm
5 GI2MO ideastream (2014). http://www.gi2mo.org/.
6 Lecce 2019 (2013) Reinventing Eutopia, Application for the title of European Capital of Culture, September 2013 available at http://www.lecce2019.it/2019/bidbook.php

web/mobile application able to facilitate the matching of demand and supply of unsold food. The execution and monitoring phase is in progress.

Table 1. Detailed data of the idea collection process.

Time Running		General	
Time since first idea posted	1 years 6 months	Number of idea	2248
Time since last idea posted	1 months	Number of idea contests	9
Time since last contest created	1 years 6 months	Total users	1226
Time since last comment posted	1 months	Total votes cast	328

3 Literature review

Literature on Idea Management (IM) are predominantly associated with innovation management in organizations [14]. As Baumgartner has reported, the practices on innovation management are not new and have been introduced in several organizations much before the IT systems explosion (e.g. 30-year history of innovation management in Toyota, had been always oriented on the road to the capture of novel ideas) [2]. However, what nowadays is known as the term ‘idea management’ in IT sector related, has been created in reference to systems that appeared in the late 90s [20]. In order to evaluate captured ideas precisely, Westerski et.al have tried to resolve the problem by introducing annotation of ideas through which the characteristics of ideas can be described highlighting their distinctive features. Reviewing IT related literature remarks the development of IM dealing with applications of IMS. Xie and Zhang, for instance, have designed an IMS to support the process of idea generation, evaluation, improvement and implementation [28]. The work of Westerski et al. [26] deals with the development of IMS and extends it from being nothing more than a box where employees could submit their ideas on a piece of paper to the web 2.0 techniques. Such transformation allows complex submission of data and data handling in IMS. They also suggests the use of semantic web principles to link organizational systems for better idea assessments [27]. IMS can also be considered as a sharing point among users and organizations [22], besides, in this manner it can be utilized as a managing and controlling tool for open innovation [11]. An example of Idea management System is OpenIDEO[7] that enables people to collaborate in developing innovative solutions to press social and environmental challenges. Idea Management System can be defined therefore as a process of needs recognition and ideas generation and evaluation [23] [16]. Those platforms aim to aid all aforementioned practices of idea management and allow organizations to collect community ideas during enterprise procedures [25]. The main contribution of this paper is to develop an approach based on idea life cycle which uses the concept of open innovation and to apply it in the context of Public Administration in order to co-create innovative public services. In this approach, all steps of life cycle are supported by the Idea Management System that interacts

[7] https://openideo.com/about-us

through a number of technological and methodological tools to facilitate collaboration and co-creation.

4 Conceptual Framework

The proposed idea life-cycle is characterized of the following six steps (Figure 1). Each step is carried out in collaboration with citizen or between citizen and public administration. It is characterized by tools that allow the responsible of each step to perform the functions in a collaborative way. IMS, starting from designed process in BPM, gives users the opportunity to create a social network where they can share, vote and promote ideas. This environment is designed around local government and citizen needs and it provides an engagement approach more efficient and effective than the usual BPM interfaces.

Idea Generation. This is the phase of ideas input from users. It can take place via two techniques: Push (the ideas about particular topics are required from public administrator) and Pull (citizen can suggest ideas to Public Administration). The actors involved are Public Administration and citizen. The importance of this phase is the free expression of citizen able to generate ideas of public and common utility and to encourage service co-creation and the participation to "*res nostra*".

Idea Management System supports the idea collection, contest creation and allows the idea sharing on most important social networks in order to encourage discussion and the promotion of the IMS. Tags and categorization of ideas allow simplifying the idea organization and research.

Idea Improvement. This is the collaboration phase and collective development of ideas. Once generated, the ideas are shared and improved thanks to the continuous collaboration between the users, who may contribute to the enrichment of ideas with comments, pictures, links, etc. In this way, from one or more initial ideas a process of co-creation, socialization and exchange of experience and knowledge is triggered. Ideas are made available to the whole community that collaborates to transform them into a structured project. Therefore the community, properly supported, can improve ideas, exploiting know-how and multiple perspectives emerging from the system. To encourage the engagement of the citizen and to create participatory behavior, gamification tools were developed.

Idea Selection. This phase supports the evaluation, selection and ranking of Ideas. Idea Management System allows to vote for the ideas that leads to a ranking. This ranking points out ideas with greater priority or the ones considered by users to be better than others. The indicators used for the evaluation are, for example: the number of threads or the vitality index that expresses how the idea remains active over time. In addition, it is possible to make even an indirect analysis of ideas through sentiment analysis that allows identifying the issues particularly important for the citizen/user. The output of this phase is the selection of those ideas to be analyzed in detail by studying the sustainability. Charts show the most popular ideas and suggest the most active members of the community. In addition to Idea Management System, a tool of sentiment analysis and a dashboard that shows (to both the PA and citizens) intuitively the data collected has been implemented.

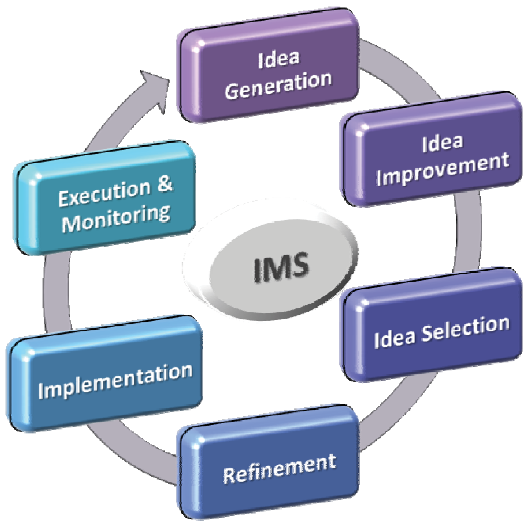


Fig. 1. Main steps of idea's life-cycle

Refinement. In this phase the selected ideas are refined thanks to the involvement of expert users (citizens or employees of PA) able to describe in detail all the steps and evolutionary processes that accompany the same idea. The expert group is formed through social office tools, by comparing the roles specified by the author of the idea and the co-authors, as necessary for the execution and implementation of the idea itself. The skills available in the profiles of each user are entered during Idea Management System registration. To assess the social and economic sustainability of the idea, is develop a methodology[12] based on Value Network Analysis [1] supported by simulation tools. The output of this phase is the transformation of the idea into a sustainable product/service in technological, economic and social terms. So, it is important to identify the role of the user, through the tools of the social office.

Implementation. The actors involved in this phase are both the PA and the citizens, experts and not. When the ideas require the development of an application and/or an information service, IMS provides a collaborative tool, allowing the user to report the needs useful to develop a service. This notification supplies with documentation and models created using the tool. The technologist will try to implement the new service by the integration of existing applications in marketplace. Where it is necessary to develop a new application that is not in the marketplace, the tool allows the user to report these needs to technologist. During each phase, in order to engage and encourage users to continue their collaboration, IMS allows both the collaborative resolution of problems, emerged during the implementation of the idea. Moreover, IMS transparently associates additional information to each phase as follows: Update on the status of implementation of the idea; Resources (technical, human ...) associated with implementation of the ideas; Information about any problems encountered in the implementation phase; Financial data; Timelines.

Execution and Monitoring. The final stage of the process of co-creation is to run the service and continuously monitor the results. The monitoring phase is very important because it allows evaluating and monitoring the success or failure of the new service through the feedback received from users and the PA. Monitoring techniques are questionnaires, interviews, surveys, reviews and feedback, collected on Idea Management System. Also in this phase, sentiment analysis tools are used. The functionality of "Analysis, filtering and tracking of ideas" provides statistics and graphs that depict the performance of the Idea Management System over time. All contents of the system in the form of a summary table and the frequency of interactions within the community can be displayed. The feedback of the users and the data collected, allow to generate suggestions for improvement and new ideas that will reopen the cycle.

5 Results and Future Developments

The proposed approach is used in the context of Public Private Partnership for a charitable cause. This need was expressed by citizens through the IMS platform and has been taken into consideration by the Local Government. The idea was to create a ''food bank'' for collecting the excess food. Based on this idea, a specific platform, which enables both donators and poor citizens to interact, has been developed. Such system reduces the food waste and, at the same time, increases the support for needy citizens. A more user-friendly interface and a mobile version could be valuable additions. A new extension, called "Social sentiment index'' is currently under development. This new extension aims at integrating the potential of sentiment analysis to identify the greatest interest of the community. However, the usage of an Idea Management System to support strategic planning in an open environment, such as urban areas, introduces a problem: administrators need further tools to prioritize efficiently interventions in the urban context. For this reason, we are working to extend the capabilities of the Idea Management System by introducing an algorithm that could calculate the user participation. The Social sentiment index will be calculated from a set of input parameters, resulting not only from the Idea Management System, but also by means of the major social networks like Facebook, Twitter, Google+ and LinkedIn. On the other hand, sentiment analysis tools, using specific algorithms as well as semantic function, will have the purpose to simplify and to categorize contents. Founded on the concept of interoperability, the project proposes a number of solutions using metadata and providing new methods of evaluation: metrics based on opinion mining, taxonomy and categorization of innovation, as well as metrics based on reports of the idea.

References

1. Allee V (2002) A Value Network Approach for Modeling and Measuring Intangibles. Transparent Enterp. Madrid
2. Baumgartner JP (2004) Big and little innovation, Report 103. 27.
3. Berkhout AJ, Hartmann D, van der Duin P, Ortt R (2006) Innovating the innovation process. Int J Technol Manag 34:390–404.

4. Boyle D, Harris M (2009) The Challenhe of Co-production.
5. Chesbrough HW (2006) The Era of Open Innovation. Manag Innov Chang 127:34–41.
6. Chesbrough HW (2006) Open Innovation: The New Imperative for Creating and Profiting from Technology. Harvard business school press
7. Chesbrough HW, Vanhaverbeke W, Eds WJ (2006) Open Innovation: Researching a New Paradigm. Oxford university press
8. Collm A, Schedler K (2012) Managing Crowd Innovation in Public Administration. Int Public Manag Rev 13:1–18.
9. Corlosquet S, Delbru R, Clark T, Polleres A, Decker S (2009) Produce and Consume Linked Data with Drupal. Semant. Web - ISWC 2009. Springer Berlin Heidelberg, pp 763–778
10. Edelmann N, Höchtl J, Sachs M (2012) Collaboration for Open Innovation Processes in Public Administrations. In: Charalabidis Y, Koussouris S (eds) Empower. open Collab. Gov. Springer Berlin Heidelberg, pp 21–37
11. Enkel E, Gassmann O, Chesbrough HW (2009) Open R&D and open innovation: exploring the phenomenon. R&D Manag 34:311–316.
12. Giangreco E, Marasso L, Chetta V, Fortunato L, Perlangeli C (2014) Modeling tools of service value networks to support social innovation in a Smart City. Vol. 21 Electron. Gov. Electron. Particip. IOS Press Ebooks, pp 206–2015
13. Hilgers D, Ihl C (2010) Citizensourcing: Applying the concept of open innovation to the public sector. Int J Public Particip 4:67–88.
14. Jensen ARV (2012) A literature review of idea management. 9th Nord. Conf. 22-24- Aarlborh Univ. Denmark
15. Koch G, Füller J, Brunswicker S (2011) Online crowdsourcing in the public sector: How to design open government platforms. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 6778 LNCS:203–212. doi: 10.1007/978-3-642-21796-8_22
16. Maraso L, Giangreco E, Storelli D, Chetta V, Camilo A (2014) Idea Management System for Smart City Planning. Interdiscip Stud J 3:227–236.
17. Mulgan G, Tucker S, Ali R, Sanders B (2007) Social Innovation what it is, why it matters and how it can be accelerated.
18. O'Reilly T (2006) What Is Web 2.0. O'Reilly Media, Inc."
19. Ramaswamy V (2009) Co-Creation of Value – Towards an Expanded Paradigm of Value Creation. Rev Lit Arts Am 11–17.
20. Rozwell C, Harris K, Caldwell F (2002) Survey of Innovative Management Technology. Gart Res (Research Note: No. M–15–1388).
21. Seidel CE, Thapa BEP, Plattfaut R, Niehaves B (2013) Selective Crowdsourcing for Open Process Innovation in the Public Sector – Are Expert Citizens Really Willing to Participate? 7th Int. Conf. Theory Pract. Electron. Gov. ACM, New York, NY, USA, pp 64–72
22. Tung W-F, Yuan S-T, Tsai J-R (2009) A custom collaboration service system for idea management of mobile phone design. Hum Factors Econ Manuf Serv Ind 19:495–509.
23. Vabdenbosch B, Saatcioglu A, Fay S (2006) Idea Management_ A Systemic View. J Manag Stud 43:259–288. doi: 10.1111/j.1467-6486.2006.00590.x
24. Van de ven A, Scott Poole M (1990) Methods for Studying Innovation Development in the Minnesota. Organ Sci 1:313–335.
25. Westerski A, Dalamagas T, Iglesias CA (2013) Classifying and comparing community innovation in Idea Management Systems.pdf. Decis Support Syst 54:1316–1326.
26. Westerski A, Iglesias CA, Nagle T (2011) The road from community ideas to organisational innovation. Int J Web Based Communities 7:493–506.
27. Westerski A, Iglesias CA, Rico FT (2010) A Model for Integration and Interlinking of Idea Management Systems. Metadata Semant. Res. Springer Berlin Heidelberg, pp 183–194
28. Xie L, Zhang P (2010) Idea Management System for Team Creation. J Softw 5:1187–1194.

Toward Advanced Visualization Techniques for Conceptual Modeling

Jens Gulden[1] and Hajo A. Reijers[2]

[1] University of Duisburg-Essen,
Universitätsstr. 9, 45141 Essen, Germany
jens.gulden@uni-due.de

[2] VU University Amsterdam,
De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands
h.a.reijers@vu.nl

Abstract. Conceptual models and their visualizations play an important role in the in the Information Systems (IS) field. Their track record, however, is mixed. While their benefits are clearly perceived, practitioners also struggle with their use. This paper picks up on a potential factor that limits the effectiveness of conceptual models, namely the poor design rationale behind their visual appearance. We argue for the benefits of a holistic view on the visual side of a conceptual modeling technique, which should draw from both perceptual and cognitive theories to improve the representation of objects. We present concrete activities and outline their fundamentals in the form of a research agenda.

Keywords: Visualization, Analysis, Modeling, Cognitive Efficiency, Human-Computer-Interaction

1 Introduction

The IS discipline is concerned with the processing and usage of information within organizational contexts. To understand and communicate the setting that information systems need to operate in, as well as to specify the requirements such systems need to adhere to, *visual conceptual models* have become an important aid. Visual conceptual models cover a wide variety of representations, such as class diagrams, business process models, use case models, etc. They are commonly used to reflect a designer's understanding of a system as it is working or as it is intended to be working. A conceptual model can be used to validate such an understanding with business professionals, or to guide developers to actually build a system, among other purposes. The effort that is spent on creating a conceptual model often pays off in terms of the efficiency or effectiveness of the project it is used in [14, 24].

Conceptual models are often displayed in visual form, e. g., as diagrams, to represent abstract data. The idea behind this is that such a *visual* approach stimulates natural characteristics of visual processing in human cognition. Despite the obvious relevance of the visual side of a conceptual model, existing research on this aspect of conceptual models is fairly thin. The exact way how to visualize information is still widely regarded as a cosmetic concern, secondary to the *meaning* of the information captured. Consider, for example, the appearance of a modeling language like YAWL: The visual appearance of the modeling constructs has obviously been a much lesser concern than the specification of their formal semantics [4]. While recent research has certainly picked up on visual aspects of a conceptual model, e. g. in [15, 9, 5], the emphasis is on the effective *use* of the visual aspects of a modeling technique – not the proper *design* or *redesign* of the technique itself.

In this paper, we argue that the visualization rationale behind a conceptual modeling technique must be treated as a primary concern by the IS field. Specifically, we believe that there is a need for *theoretic strengthening*, which should rest on a deeper understanding of what conceptual models aim to capture: the complex socio-technical interplay among humans and software systems.

Our contribution is a research agenda which is founded on the insight that the capabilities of the human brain are essential ingredients for visual design. The research that we outline is meant to enhance the use of conceptual models toward a higher level of effectiveness. Despite their noted success, it must be acknowledged that conceptual models still pose difficulties with users, specifically non-experts [1, 3, 19]. By considering both the conceptual and perceptual qualities of IS artifacts in an integrated way, a conceptual model potentially becomes a more readily usable asset. A key element to achieve this is to assign a real-world semantics to its elements, which was already identified as a research challenge in [24] but not picked up on so far.

This position paper is structured as follows. In Section 2, theories and related work will be discussed that are relevant for our research agenda. Proposals for research directions are then presented in Section 3. We will conclude this paper with a summary and outlook in Section 4.

2 A theoretical view on information visualization

2.1 Demand for theoretical underpinnings of information visualization in IS

While visualization techniques are to some extent studied by IS scholars, we consider it remarkable that the techniques for conceptual modeling have barely evolved from a visual perspective. The most dominant techniques for business process modeling, for example, are still highly similar to the archetypical "process flow chart" proposed in the 1920s [7]. Specifically, a process model – whether expressed as EPC, BPMN model or UML Activity Diagram – is still shown as a static diagram in which different types of symbols are connected by arrows. Considering advances that have been made in fields such as computer animation and information visualization in recent years, it seems reasonable to expect that untapped sources exist for better depictions of conceptual models. In this section, we will review a number of disciplines and their corresponding theories for inspiration.

Graphic Design The field of Graphic Design research has developed and explicated design recommendations and quality criteria for visualizations in diverse areas of practical applications [18]. The focus is primarily on aesthetics, with questions about the cognitive impact of visual expressions in the center of the discipline, and reasoning about transporting conceptual meaning and understanding towards its periphery. Graphic Design offers a stable core of knowledge about the effectiveness of different visualization modes. The terminological apparatus of Graphic Design research for reflecting about visualizations goes far beyond the simplified notion of hierarchically composed, atomistic graphical shapes as being common in IS today. It thus can be expected that incorporating parts of the stable core knowledge achieved in Graphic Design will provide a relevant theoretic advance for IS research on conceptual model visualization.

Interaction Design Research activities in Interaction Design especially care about how environments for presenting information should be shaped in order to allow humans to accurately make sense out of information [20]. The core idea is that suitable ecological settings influence the way information is perceived and processed [23]. Interaction Design does not embrace visualization techniques as its core concern, but rather cares for analyzing information needs and reasonably arranging visualization techniques and other means of human-to-software communication to fulfil them efficiently. In this sense, it lies in the very center of Interaction Design's interests to theorize about relationships between perceiving and understanding, which is one key element for an advanced handling of visualizations which is yet missing in IS research on visualization.

Cognitive Science Cognitive Sciences operate in the force field between reasoning about thinking and understanding, and examining characteristics of the mind, down to the physical functioning of the human brain. While in Graphic Design phenomena such as gestalt perception and pre-conceptual categorizations are taken as-is and are systematized specifically for the purpose of providing design recommendations for visualizations, the focus in Cognitive Science is much wider and incorporates research about all cognitive phenomena and mental capabilities [2]. Besides language use, memorizing, and reasoning, e. g., this also covers visual processing of the human mind and the relationship between seeing and understanding. When it comes to finding means for assessing the efficiency of specific modes of visualizations for conceptual models in IS, Cognitive Sciences are likely to provide appropriate fundamentals.

Philosophy of Mind The philosophical direction of Embodied Cognition offers an explanation model for human thinking which derives capabilities, such as speaking languages or understanding abstract concepts, from basic experiences humans make as bodily beings in a physical world [12]. According to this approach, humans repeatedly perceive patterns (such as "things fall down when they don't have a surface to rest on", or "two objects cannot be simultaneously in the same place"), which, after repeated steps of metaphorical abstractions, finally let higher-level cognitive capabilities emerge. Given the elaborations of this idea available from philosophical works, a translation of key concepts for application in IS can be expected to contribute to a rich terminological apparatus for reflection on model visualization.

2.2 Related work

As stated in the introduction, most work that considers visual aspects of conceptual models focuses on the best possible use of the visual elements of existing techniques, as in [15, 9, 5]. In these types of work, the modeling technique itself is not put into question. Only a few attempts to systematically introduce research questions around model visualizations exist in IS and its neighboring discipline computer science. [22] attempts to develop a measure for the economic value of visualization based on effectiveness and efficiency. By focussing only on cost aspects, however, a qualitative evaluation of information visualization is not considered at all. As [17] remarks, the differences between textual languages and visual representations are so diverse "that fundamentally different principles are required for evaluating and designing visual languages". The proposed solution, however, remains limited by typical paradigmatic presuppositions that are common in the way visualization is treated in IS research today (see 2.3).

Other contributions in the literature, which put data or information visualization into focus, e. g. [21, 13, 25], operate mainly on a visual design level and do not embed their examinations into the wider focus of reasoning about developing modeling techniques and improved information systems.

2.3 Shortcomings of current approaches

The current state of theoretic reflection about model visualization is characterized by a set of paradigmatic limitations, which narrow down existing approaches to operate with restricted notions of visualizations. Some presuppositions by which current research is constricted become visible in Moody's attempt [17] to provide a theory on the "physics" of notation. While the approach is thoroughly motivated by the identification of severe weaknesses in current reflections on visualizations, the examination carried out contains statements such as "there must be a one-to-one correspondence between symbols and their referent concepts" (cited from [8]). This explicitly excludes a notion of patterns as carriers of meaning in visualizations, although there is strong evidence that exactly the cognitive capabilities of processing of patterns rather than linear language is one key element of understanding visual perception.

Additionally, [17] claims that it "says nothing about [. . .] semantic issues". This expresses an aim for introducing a methodological simplification in order to make the examination better handleable. However, as argued above, there are good reasons to believe that fruitful answers to questions about predicting and measuring the cognitive effectiveness and efficiency of visualizations can only be given when a joint theoretic view on conceptual thinking and perceptual processes is taken in. Indeed, the methodological separation of conceptualization and representation might be the *very* reason why visualization research in IS currently seems to be stuck in a crisis.

3 Research direction

3.1 Seeing is thinking, and vice versa – overcome the methodical reductionism that divides conceptualization and perception

It is a widespread belief that the use of visualization for communication increases the efficiency and accuracy of communication, because the cognitive apparatus of humans handles visual impressions differently from spoken or written language. When seeing, humans can simultaneously grasp an unvisualizationderstanding on multiple levels of granularity. Complex relationships can easily be understood by being affected by patterns, and understanding visual impressions can happen in parallel. This is only possible because in our minds the *conceptual qualities* transported by a visualization, i. e., knowledge expressed by it, and the *perceptual qualities*, which are the visual impressions that shape the way our minds handles the perceived impressions, are intricately intertwined [11, 16].

The very question of how to make "good" visualizations implies that conceptual thinking and visual thinking must be brought together. Separating conceptualization and perception would lead, in our view, into the wrong methodological direction. By this separation, any attempt to gain a prescriptive theoretic approach that allows to consciously judge why some visualizations will lead to a more efficient and accurate understanding than others is jeopardized from the very beginning, because the "understanding" side is excluded beforehand.

Rather than trying to approach the research questions by seemingly reducing the complexity, but, in fact, preventing an appropriate argumentation architecture for answering the questions, a theory seems required that offers a combined view on conceptual semantics and perceptual qualities. The road toward an elaboration of such an approach is sketched in the remaining part of this paper.

3.2 Developing an advanced theory for information visualization

We suggest to follow some coarse-grained steps for performing research in the direction that we outlined.

In an initial phase, the existing body of knowledge from disciplines such as graphic design, interaction design, cognitive sciences, gestalt psychology, and philosophy of mind should be examined to find stable theoretic knowledge in which conceptual elements and visual design rationales are systematically combined. Prospectively, it will be sufficient to concentrate on core elements of the respective disciplines which are no longer undergoing intra-disciplinary discussions.

Based on conceptualizations from these research areas, a linkage to IS can be established by applying its methodical means, such as conceptual modeling, meta-modeling, knowledge representation and transformation techniques, to "translate" the imported knowledge from other disciplines into languages of IS. This procedure resembles one of the very core competencies and purposes of IS, namely the terminological reconstruction of domains of discourse to (semi-)formal languages [6] for the further design and development of information systems. The notable difference in this case is the terminological reconstruction performed on the methodological level to establish scientific means for constructing IS methods, instead of performing it on the methodical level, where external domains of discourse are investigated.

Once conceptualizations in the languages of IS are made explicit, software can be described using the imported concepts. For research purposes, tool support for software-based visualization methods in IS can be provided. The capabilities of these solutions cannot yet be described more precisely, because they will become characterizable only as a result of the examinations on imported knowledge from other disciplines.

With the existence of prototypical methods with implemented tooling support, visualization approaches will become eligible for systematic evaluation through established design science criteria [10]. Evaluation can either be performed qualitatively, i. e., by analyzing and comparing visualization approaches with the terminological equipment developed through the reconstruction of imported theories, or quantitatively, i. e., by applying empirical experiments and surveys.

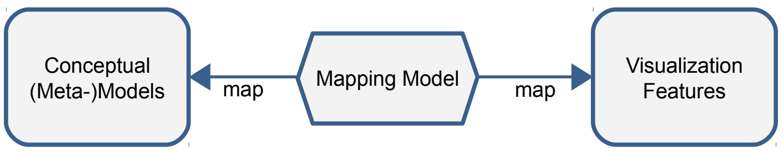
3.3 Theory architecture

Our proposal extends the existing and in our view oversimplified architecture of existing visualization approaches for conceptual models, as it is displayed in Fig. 1 (a). The traditional view assumes that visualizations can sufficiently be described by one-to-one mappings between conceptual elements and visual representations. The specification of visualizations is performed on the level of meta-models, where types of model elements and types of visual elements are related to each other.

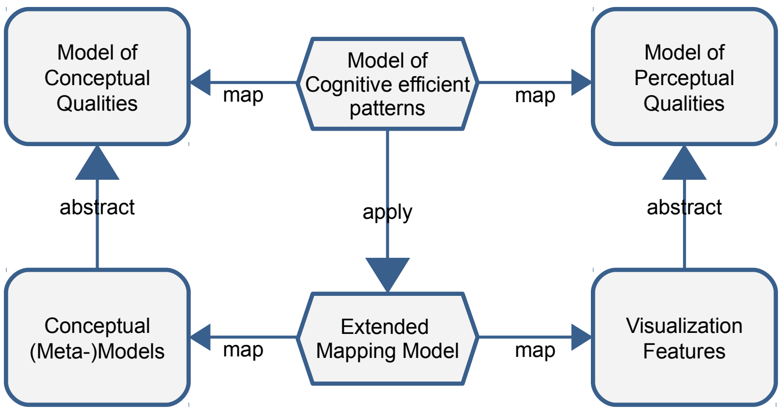
The extensions of the theory architecture that our proposed research program implies incorporate an *additional* level of reflection on the meta^2 level. Fig. 1 (b) illustrates this by repeating the original mapping structure in its lower part, and adding the element "Model of Conceptual Qualities" which represents terminology to describe characteristics of meta-concepts, the element "Model of Perceptual Qualities" which represents imported knowledge about visualizations with regard to their cognitive impact and features that influence their understanding, and the element "Model of cognitive efficient patterns" which stands for IS-specific insights about combing the other two. It should be noted that at the current stage of formulating the demands for a research program on conceptual model visualization, the introduced meta elements merely act as placeholders for research results yet to be achieved. Possible manifestations of these elements, and their representations in formal document artifacts, still remain to be elaborated.

4 Conclusion

In this work, we have signalled a fundamentally flawed view on the role of visualization aspects of conceptual models. Our argument, inspired by a range of



(a)



(b)

Fig. 1: Traditional notion of a visualization specification (a), and the suggested extensions (b)

theories, is that a separation of conceptual thinking and perceptual processes is a dead end. Considering the importance of conceptual modeling for the IS field, it seems appropriate for researchers in this discipline to take on the challenge and embrace a wider perspective on visualization research than characteristic for the state of the art. It is our hope that the research agenda we provided may serve as an inspiration for this endeavor.

References

1. Dinesh Batra and George M Marakas. Conceptual data modelling in theory and practice. *European Journal of Information Systems*, 4(3):185–193, 1995.
2. José Luis Bermúdez. *Cognitive Science: An Introduction to the Science of the Mind*. Cambridge University Press, Cambridge, 2nd edition, 2014.
3. Islay Davies, Peter Green, Michael Rosemann, Marta Indulska, and Stan Gallo. How do practitioners use conceptual modeling in practice? *Data & Knowledge Engineering*, 58(3):358–380, 2006.
4. Kathrin Figl, Jan Mendling, Mark Strembeck, and Jan Recker. On the cognitive effectiveness of routing symbols in process modeling languages. In *Business Information Systems*, pages 230–241. Springer, 2010.

5. Kathrin Figl and Mark Strembeck. On the importance of flow direction in business process models. In *ICSOFT-EA 2014 - Proceedings of the 9th International Conference on Software Engineering and Applications, Vienna, Austria, 29-31 August, 2014*, pages 132–136, 2014.
6. Ulrich Frank. Multi-level modeling - toward a new paradigm of conceptual modeling and information systems design. *Business & Information Systems Engineering (BISE)*, 6(3), 2014.
7. Frank B Gilbreth and LM Gilbreth. Process charts and their place in management. *Mechanical engineering*, 70:38–41, 1922.
8. Nelson Goodman. *Languages of Art: An Approach to a Theory of Symbols*. Hackett Publishing, Indianapolis, 1968.
9. Thomas Gschwind, Jakob Pinggera, Stefan Zugal, Hajo A Reijers, and Barbara Weber. A linear time layout algorithm for business process models. *Journal of Visual Languages & Computing*, 25(2):117–132, 2014.
10. Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, March 2004.
11. E. Husserl and D. Moran. *Logical Investigations*, volume 2 of *International library of philosophy and scientific method*. Routledge, 2001.
12. G. Johnson, M.; Lakoff. *Philosophy in the Flesh : The Embodied Mind and Its Challenge to Western Thought*. Basic Books, New York, 1999.
13. Andy Kirk. *Data Visualization: a successful design process*. Packt Publishing, Birmingham, 2012.
14. Ned Kock, Jacques Verville, Azim Danesh-Pajou, and Dorrie DeLuca. Communication flow orientation in business process modeling and its effect on redesign success: Results from a field study. *Decision Support Systems*, 46(2):562–575, 2009.
15. Marcello La Rosa, Arthur HM Ter Hofstede, Petia Wohed, Hajo A Reijers, Jan Mendling, and Wil MP van der Aalst. Managing process model complexity via concrete syntax modifications. *Industrial Informatics, IEEE Transactions on*, 7(2):255–265, 2011.
16. M. Merleau-Ponty. *Phenomenology of Perception*. Routledge classics. Routledge, 2002.
17. Daniel L. Moody. The “physics” of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779, 11/12 2009.
18. R. Poulin. *The Language of Graphic Design: An Illustrated Handbook for Understanding Fundamental Design Principles*. Rockport Publishers, 2011.
19. Michael Rosemann. Potential pitfalls of process modeling: part b. *Business Process Management Journal*, 12(3):377, 2006.
20. Dan Saffer. *Designing for Interaction: Creating Innovative Applications and Devices*. New Riders Publishing, Thousand Oaks, 2nd edition, 2009.
21. Robert Spence. *Information Visualization*. Prentice Hall, Upper Saddle River, 2nd edition, 2007.
22. J.J. van Wijk. The value of visualization. In C. Silva, E. Groeller, and H. Rushmeier, editors, *Proceedings IEEE Visualization 2005*, pages 76–86, 2005.
23. K. J. Vicente and J. Rasmussen. Ecological interface design: Theoretical foundations. *IEEE Transactions on Systems, Man and Cybernetics*, (22):589–606, 1992.
24. Yair Wand and Ron Weber. Research commentary: information systems and conceptual modeling – a research agenda. *Information Systems Research*, 13(4):363–376, 2002.
25. Hunter Whitney. *Data Insights: New Ways to Visualize and Make Sense of Data*. Morgan Kaufman, Waltham, 2013.

Towards Visually Monitoring Multiple Perspectives of Business Process Compliance[⋆]

David Knuplesch[1], Manfred Reichert[1], and Akhil Kumar[2]

[1] Institute of Databases and Information Systems, Ulm University, Germany
{david.knuplesch,manfred.reichert}@uni-ulm.de
[2] Smeal College of Business, Pennsylvania State University, PA, USA
akhilkumar@psu.edu

Abstract. A challenge for enterprises is to ensure conformance of their business processes with imposed compliance rules. Usually, the latter may constrain multiple perspectives of a business process, including control flow, data, time, resources, and interactions with business partners. Like in process modeling, visual languages for specifying compliance rules have been proposed. However, business process compliance cannot be completely decided at design time, but needs to be monitored during run time as well. This paper introduces an approach for visually monitoring business process compliance. In particular, this approach covers all relevant process perspectives. Furthermore, compliance violations cannot only be detected, but also be visually highlighted emphasizing their causes. Finally, the approach assists users in ensuring compliant continuations of a running business process.

Keywords: business process compliance, compliance monitoring

1 Introduction

Correctness issues of business process models have been intensively studied for more than a decade. While early work focused on syntactical correctness and soundness (e.g., absence of deadlocks and lifelocks), recent approaches have focused on how to ensure the compliance of business processes with semantic constraints. Usually, respective *compliance rules* stem from domain-specific requirements, like, for example, corporate standards or legal regulations [1], and need to be ensured in all phases of the process life cycle [2].

In this context, approaches addressing the compliance of running business process instances are covered by the notion of *compliance monitoring* [3–5]. In general, events of running process instances need to be considered to detect and report run-time violations of compliance rules (cf. Fig. 1). Thereby, *reactive* and *proactive* monitoring need to be distinguished. Regarding the former, compliance violations are reported once they have occurred. In turn, proactive monitoring

[⋆] This work was done within the research project C^{3}Pro funded by the German Research Foundation (DFG) under project number RE 1402/2-1.

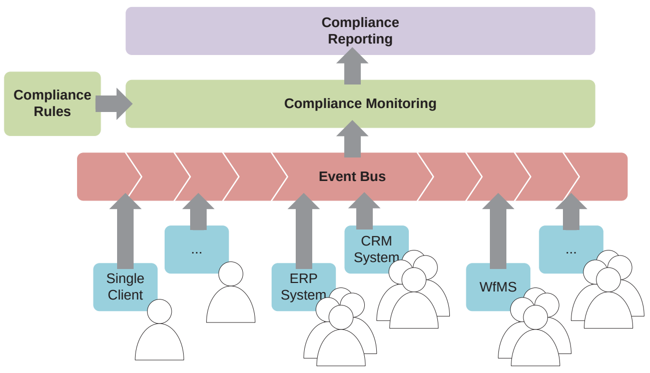


Fig. 1: Compliance Monitoring [7, 9]

aims to prevent compliance rule violations; e.g., by suggesting appropriate tasks that still need to be executed to meet a compliance rule. While early approaches for monitoring compliance focused on the control flow perspective, more and more, additional process perspectives have been considered as well (e.g. [6]). In particular, the data, resource and time perspectives as well as interactions with business partners have been addressed. Other advanced work has dealt with the traceability of compliance violations [7, 8]. However, existing approaches do not provide a satisfactory solution that combines an expressive compliance rule language with full traceability [9].

As example consider the event log from Fig. 2 that refers to an order-to-

#	date	time	type	id	details
⋮					
37	1/7/2013	15:27	receive	124	Request
38	1/7/2013	15:27	write	124	customer = Mr.Smith
39	1/7/2013	15:27	write	124	amount = 15.000€
39	1/7/2013	15:27	end	124	Request
⋮					
55	1/7/2013	18:03	receive	592	Request
56	1/7/2013	18:03	write	592	customer =Mrs.John
57	1/7/2013	18:03	write	592	amount = 27.000€
58	1/7/2013	18:03	end	592	Request
⋮					
77	2/7/2013	15:43	start	234	SolvencyCheck (Mrs. Brown)
78	2/7/2013	15:43	read	234	customer = Mr.Smith
79	2/7/2013	15:54	write	234	rating= high
80	2/7/2013	15:55	end	234	SolvencyCheck
⋮					
91	2/7/2013	18:13	start	453	Approval (Mr. Muller)
92	2/7/2013	18:14	read	453	customer = Mr.Smith
93	2/7/2013	18:14	read	453	rating = high
94	2/7/2013	18:17	write	453	result= granted
95	2/7/2013	18:18	end	453	Approval
96	2/7/2013	18:19	start	642	Approval (Mrs. Brown)
97	2/7/2013	18:20	read	642	customer = Mrs.John
98	2/7/2013	18:23	write	642	result = granted
99	2/7/2013	18:23	end	642	Approval
⋮					

Compliance rules
c$_1$ When a request item with an amount greater than 10,000 is received from an agent, the request must not be approved unless the solvency of the respective customer was checked. The latter task must be started at max three days after the receipt. Further, task approval and task solvency check must be performed by different staff members.
c$_2$ After approval of a request item, the agent must be informed about the result within one days.
c$_3$ After starting the production related to a particular order the latter may only be changed by the head of production.

Fig. 2: Event log of order-to-delivery processes and compliance rules

delivery process. Compliance rule c_1, which is shown on the right, is satisfied in one case, but violated in another. In particular, the depicted log refers to two different request items related to customers *Mr. Smith* and *Mrs. John.* These items, in turn, trigger two different instances of compliance rule c_1. In both cases, the amount is greater than 10,000 € and hence a solvency check is required. However, the latter was only performed for the request item of Mr. Smith, but not for the one of *Mrs. John* (i.e., c_1 is violated in the latter case). Besides the violation of c_1, compliance rule c_2 is violated twice as well. While the violated instance of c_1 can never be successfully completed, the violations of c_2 still can be healed by informing the *agent.* The rule examples further indicate that solely monitoring control flow dependencies between tasks is not sufficient to ensure compliance at run time. In addition, constraints in respect to the data, time and resource perspectives of a business process must be monitored as well as the interactions this process has with partner processes [10, 11, 9]. For example, the data perspective of compliance rule c_1 is addressed by activity *request item* and its data *amount.* Receiving the *request item*, in turn, represents an interaction with a business partner. Furthermore, the phrase *by different staff members* deals with the resource perspective, whereas the condition *at maximum three days* refers to the time perspective. To meet practical demands, compliance monitoring must not abstract from these process perspectives.

This paper sketches an approach for visually monitoring multiple perspectives of business process compliance. For this purpose, we annotate the visual *extended Compliance Rule Graph (eCRG)* language [11, 12] with text, markings and symbols to highlight the current state of a compliance rule. The annotations not only indicate compliance violations, but may also be utilized for recommending the next process steps required to restore compliance. Furthermore, they allow us to clearly distinguish between fulfilled and violated instances of an eCRG. Note that the eCRG language adequately supports the time, resource and data perspectives as well as interactions with business partners.

The remainder of this paper is structured as follows: The approach for visually monitoring multiple perspectives of business process compliance is outlined in Section 2. Section 3 concludes the paper and provides an outlook on future research.

2 eCRG Compliance Monitoring

This paper utilizes the extended Compliance Rule Graph (eCRG) language for compliance monitoring [11, 12]. The eCRG language is a visual language for modeling compliance rules. It is based on the Compliance Rule Graph (CRG) language [7]. As opposed to the latter, the eCRG language not only focuses on the control flow perspective, but additionally provides integrated support for the resource, data and time perspectives as well as for the interactions with business partners. Fig. 3 provides an overview of eCRG elements, which are applied in Fig. 4 in order to model the compliance rules from Fig. 2.

In the following, we sketch the approach towards visually monitoring multiple perspectives of business process compliance at runtime. As discussed in Sect. 1,

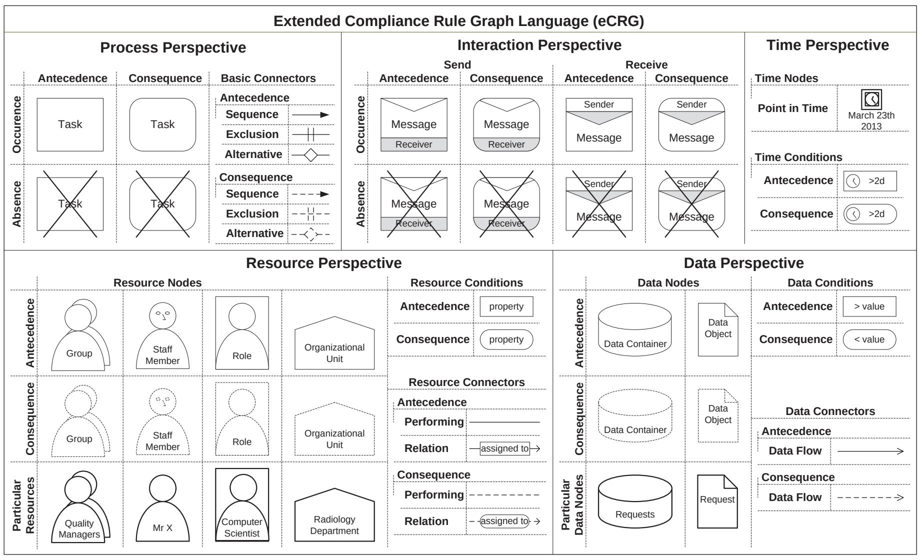


Fig. 3: Elements of the eCRG language

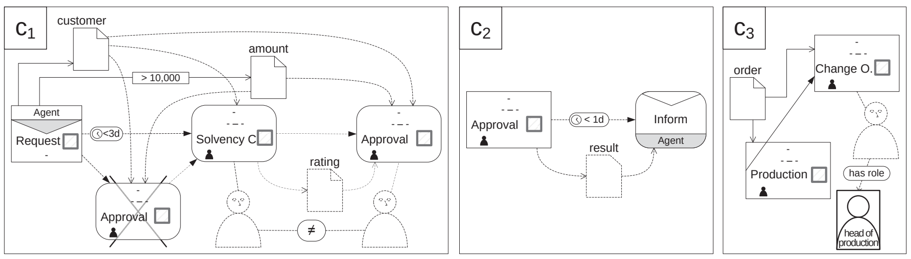


Fig. 4: Modeling compliance rules $c_1 - c_3$ with the eCRG language

compliance monitoring is based on streams of events, which occur during the execution of business processes, and aims to determine or prevent compliance violations. For this purpose, we extend the approach presented in [7] and annotate the elements of an eCRG when processing events.

Events The processing of event logs requires a well-defined set of events. As the approach enables compliance monitoring for multiple process perspectives, we not consider only events referring to the start and end of tasks, but additionally monitor data flow events as well as events that correspond to the sending and receipt of messages. Furthermore, events may include temporal information as well as information about involved resources. Table 1 summarizes the supported event types. Each event refers to the occurrence time as well as a unique id. The latter enables us to identify correlations between the start, end and data flow events of the same task or message.

Table 1: Supported Events

Task events	Message events	Data flow events
`start`(time, id, tasktype, performer)	`send`(time, id, message)	`write`(time, id, value $\xrightarrow{\text{param}}$ source)
`end`(time, id, tasktype, performer)	`receive`(time, id, message)	`read`(time, id, value $\xleftarrow{\text{param}}$ source)
	`end`(time, id, message)	

eCRG Markings To monitor the state of a compliance rule, we annotate and mark eCRG elements with symbols, colors and text (cf. Figs. 5). Such a *marking of an eCRG* results in an annotated eCRG highlighting whether or not the events corresponding to a particular node have occurred so far. Furthermore, it describes whether the conditions of edges and attachments are satisfied, are violated or have not been evaluated yet.

Event Processing We exemplarily describe how events are processed for an eCRG (cf. Fig. 6) and refer to [13] for a formal specification of the operational semantics of the eCRG langauge. First, all markings are updated to the point in time of the specific event. Second, the effects of the update (i.e., adapted annotations) are propagated to succeeding as well as skipped elements. Third, the actual *event handling* takes place depending on the type of the current event. Finally, the effects of the latter step are propagated as well.

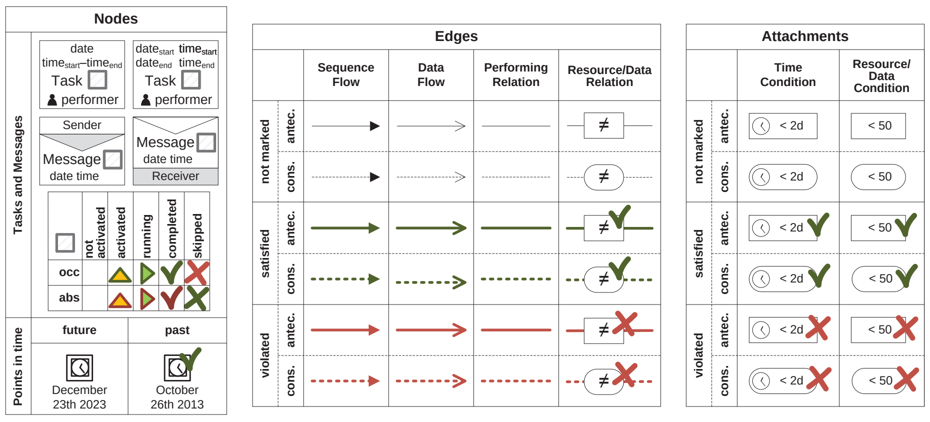


Fig. 5: Annotations of eCRG elements

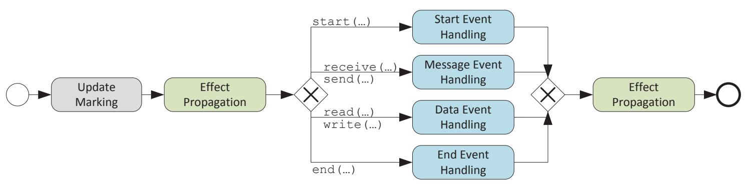


Fig. 6: Processing of start, message, data and end events

Fig. 7b illustrates the handling of a message event. In particular, the marking of the activated message node *request*, which matches the event, changes to ▶. Accordingly, the starting time is set. Start events are processed like message events, but additionally store information about the responsible actor. Start and message events are handled non-deterministically; i.e., the changes are applied to a copy of the original marking. Fig. 7c shows the handling of data events. In particular, the corresponding data flow edge is annotated with the data value passed. Fig. 7e illustrates the handling of an end event. In particular, the annotations of *request* is changed to ✓ and its ending time is set accordingly.

Fig. 7g illustrates how a marking is updated to the current point in time. In particular, the annotations of point in time nodes, which now refer to the past, change to ✓. Finally, time conditions on running task nodes or sequence flow edges are skipped (✕) if they are no longer satisfiable (cf. Fig. 7g).

According to Fig. 6, effects of events and updates must be propagated to indirectly affected eCRG elements in order to ensure correct annotations (e.g., activation of subsequent task nodes) as well as to detect contradictory annotations related to the data and resource perspectives. In particular, data values are propagated from writing and reading data flow edges to dependent data object and data container nodes (cf. Fig. 7d). In turn, resources are propagated from task nodes to dependent resource nodes via the connecting resource edges. The propagation fails, if a resource, data object or container node was set to a different value before. In this case, the respective edge is skipped (✕). Furthermore, conditions and relations are evaluated as soon as possible (cf. Fig. 7d). If any element of the eCRG corresponding to a task or message node is skipped (e.g., due to a failed data/resource propagation or a violated condition), the corresponding task or message node will be skipped as well. Then, outgoing sequence flows of completed nodes are marked as satisfied, whereas non-marked incoming edges of already started nodes are skipped. Sequence flow edges from and to skipped nodes are skipped as well. Task and message nodes, in turn, become activated when all incoming sequence flows they are depending on are satisfied. Additionally, task or message nodes will be skipped if they depend on sequence flows that were skipped as well. Note that the latter might require skipping further sequence flow edges, and so forth (cf. Fig. 7h). Finally, Fig. 7i provides a marking that fulfills c_1 for the request of Mr. Smith. In turn, Figs. 7h+7k highlight conflicts regarding time and resource perspectives.

The non-deterministic processing of start and message events may result in sets of markings. Therefore, single fulfilling or conflicting markings do not imply (non-)compliance with the related rule. For this purpose, [13] provides mechanisms to evaluate such sets and to select the most meaningful markings.

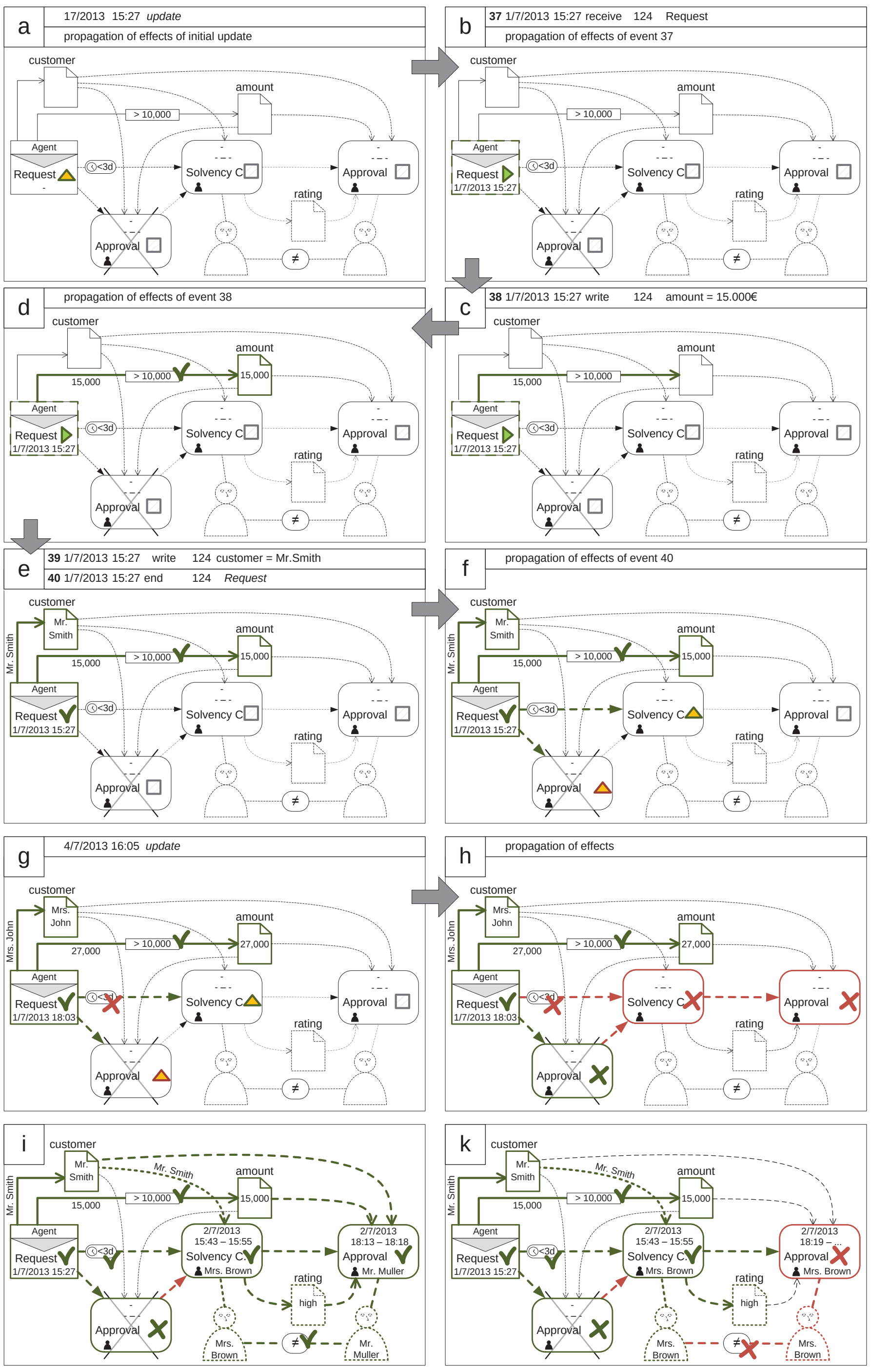

a
17/2013 15:27 update
propagation of effects of initial update
b
37 1/7/2013 15:27 receive 124 Request
propagation of effects of event 37
c
38 1/7/2013 15:27 write 124 amount = 15.000€
d
propagation of effects of event 38
e
39 1/7/2013 15:27 write 124 customer = Mr.Smith
40 1/7/2013 15:27 end 124 Request
f
propagation of effects of event 40
g
4/7/2013 16:05 update
h
propagation of effects
i
k
customer
amount
> 10,000
Agent
Request
<3d
Solvency C
Approval
rating
≠
15,000
27,000
Mr. Smith
Mrs. John
1/7/2013 15:27
1/7/2013 18:03
2/7/2013 15:43 – 15:55
2/7/2013 18:13 – 18:18
2/7/2013 18:19 – ...
Mrs. Brown
Mr. Muller
high

Fig. 7: eCRG markings and handling of events

3 Summary and Outlook

Business process compliance has gained increasing interest during the last years. Several approaches focus on compliance monitoring at run time [3–5]. However, existing approaches do not provide a satisfactorily solution combining an expressive language with full traceability [9]. This paper, therefore, has sketched an approach for visually monitoring multiple perspectives of business process compliance. For this purpose, we utilize the extended compliance rule graph (eCRG) language [11] that enables the visual modeling of compliance rules with support of the control flow, data, time, and resource perspectives as well as the interactions with partners. We annotate eCRGs with text, colors and symbols to visually highlight the current compliance state as well as to indicate its evolution during process execution. Note that we formally specified this operational semantics of the eCRG language in a technical report [13]. As opposed to existing approaches, we aim to combine full traceability with an expressive visual notation.

As next step, we will investigate algorithms for eCRG-based compliance monitoring utilizing the eCRG operational semantics we presented in [13]. Finally, we will provide a proof-of-concept implementation to evaluate the approach.

References

1. Sadiq, S., Governatori, G., Naimiri, K.: Modeling control objectives for business process compliance. In: BPM'07, Springer (2007) 149–164
2. Knuplesch, D., Reichert, M.: Ensuring business process compliance along the process life cycle. Technical Report 2011-06, Ulm University (2011)
3. Namiri, K., Stojanovic, N.: Pattern-Based design and validation of business process compliance. In: CAiSE'07, Springer (2007) 59–76
4. Alles, M., Kogan, A., Vasarhelyi, M.: Putting continuous auditing theory into practice: Lessons from two pilot implementations. Inf Sys **22**(2) (2008) 195–214
5. van der Aalst, W.M.P., et al: Conceptual model for online auditing. Decision Support Systems **50**(3) (2011) 636–647
6. Montali, M., et al.: Monitoring business constraints with the event calculus. ACM Trans. Intell. Syst. Technol. **5**(1) (2014) 17:1–17:30
7. Ly, L.T., Rinderle-Ma, S., Knuplesch, D., Dadam, P.: Monitoring business process compliance using compliance rule graphs. In: CoopIS'11. (2011) 82–99
8. Maggi, F., Montali, M., Westergaard, M., van der Aalst, W.M.P.: Monitoring business constraints with linear temporal logic: an approach based on colored automata. In: BPM'11. (2011) 132–147
9. Ly, L.T., et al.: A framework for the systematic comparison and evaluation of compliance monitoring approaches. In: EDOC'13, IEEE (2013) 7–16
10. Knuplesch, D., et al.: Towards compliance of cross-organizational processes and their changes. In: BPM'12 Workshops, Springer (2013) 649–661
11. Knuplesch, D., et al.: Visual modeling of business process compliance rules with the support of multiple perspectives. In: ER'2013, Springer (2013) 106–120
12. Semmelrodt, F., Knuplesch, D., Reichert, M.: Modeling the resource perspective of business process compliance rules with the extended compliance rule graph. In: BPMDS'14, Springer (2014) 48–63
13. Knuplesch, D., et al.: An operational semantics for the extended compliance rule graph language. Technical Report 2014-6, Ulm University (2014)

A Model-driven Method and a Tool for Developing Gesture-based Information System Interfaces

Otto Parra [1,2], Sergio España [1], Oscar Pastor [1]

[1] PROS Research Centre, Universitat Politècnica de València, Spain
[2] Computer Science Department, Universidad de Cuenca, Ecuador

otpargon@upv.es,{sergio.espana, opastor}@pros.upv.es

Abstract. Considering the technological advances in touch-based devices, gesture-based interaction has become a prevalent feature in many application domains. Information systems are starting to explore this type of interaction. Currently, gesture specifications are hard-coded by developers at the source code level, hindering its reusability and portability. Similarly, defining new gestures in line with users' requirements is further complicated. This paper describes a model-driven approach to include gesture-based interaction in desktop information systems and a tool prototype to: capture user-sketched multi-stroke gestures and transform them into a model, automatically generating the gesture catalogue for gesture-based interaction technologies and gesture-based interface source code. We demonstrate our approach in several applications, ranging from case tools to form-based information systems.

Keywords: Model-driven architecture, gesture-based interaction, multi-stroke gestures.

1 Introduction

New devices come together with new types on interfaces (e.g. based on gaze, gesture, voice, haptic, brain-computer interfaces). Their aim is to increase the naturalness of interaction [1], although this is not exempt from risks. Due to the popularity of touch-based devices, gesture-based interaction is slowly gaining ground on mouse and keyboard in domains such as video games and mobile apps. Information systems (IS) are likely to follow the trend, especially, in supporting tasks performed outside the office [2].

Several issues may hinder the wide adoption of gesture-based interaction in complex information systems engineering. Gesture-based interfaces have been reported to be more difficult to implement and test than traditional mouse and pointer interfaces [3]. Gesture-based interaction is supported at the source code level (typically third-generation languages) [4]. This involves a great coding and maintenance effort when multiple platforms are targeted, has a negative impact on reusability and portability, and complicates the definition of new gestures. Some of these challenges can be tackled by following a model-driven development (MDD) approach provided that gestures and gesture-based interaction can be modelled and that it is possible to automatically generate the software components that support them.

This paper introduces an MDD approach and a tool for gesture-based IS interface development, which is intended to allow software engineers focusing on the key aspects of gesture based information system interfaces; namely, defining gestures and specifying gesture-based interaction. Coding and portability efforts are alleviated by means of model-to-text (M2T) transformations.

2 State of art

2.1 State of the art on gesture representation

The representation of gestures according to related literature can be classified into some categories: (a) **based on regular expressions** [5] [6]: a gesture is defined by means of regular expressions formed by elements such as ground terms, operators, symbols, etc.; (b) **based on a language specification** [7]: XML is used as reference to describe gestures; (c) **based on demonstration** [8]: developers to define gestures, test the generated code, refine it, and, once they are satisfied with them, include the code in the applications.

In this research work we propose a model-driven approach in order to represent gestures with a high-level of abstraction, enabling platform-independence and reusability. By providing the proper transformations, it is possible to target several gesture recognition technologies. We focus on user-defined, multi-stroke, semaphoric gestures [9].

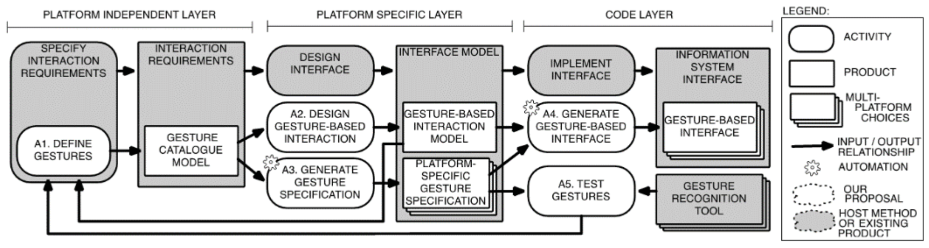
2.2 The role of gesture-based interfaces in IS engineering

Gesture-based interfaces can play two major roles in IS engineering, depending on whether we intend to incorporate this natural interaction into (i) CASE tools or (ii) into the IS themselves. In the former case, the interest is to increase the IS developers' efficiency, whereas in the latter the aim is to increase IS usability, especially in operations in the field, where the lack of a comfortable office space reduces the ergonomics of mouse and keyboard. In both cases, gesture-based interface development methods and tools are needed. Some examples of methods and tools are described in [10], [11], where the authors propose a method to integrate gesture-based interaction in an interface.

In this work, we propose a similar flow to that of [10], but automate the implementation of gesture-based interfaces by means of model transformations. In future work, we plan to provide support to the ergonomic principles by [11].

3 The gestUI method

gestUI is a user-driven and iterative method that follows the MDD paradigm. The main artefacts are models which are conform to the Model-Driven Architecture, a generic framework of modelling layers that ranges from abstract specifications to the software code (indicated at the top of Fig. 1).



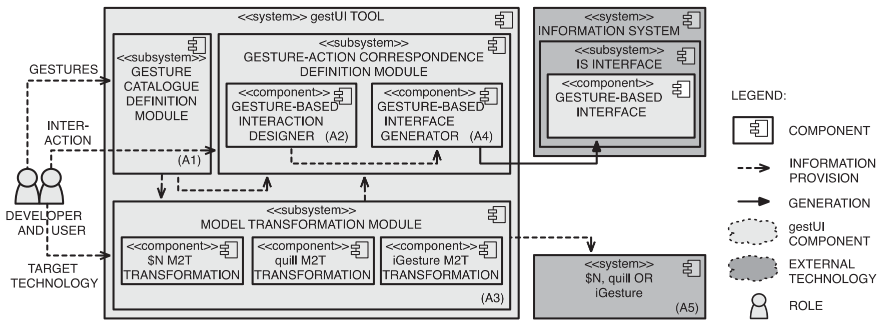
PLATFORM INDEPENDENT LAYER
PLATFORM SPECIFIC LAYER
CODE LAYER
LEGEND:
SPECIFY INTERACTION REQUIREMENTS
A1. DEFINE GESTURES
INTERACTION REQUIREMENTS
GESTURE CATALOGUE MODEL
DESIGN INTERFACE
A2. DESIGN GESTURE-BASED INTERACTION
A3. GENERATE GESTURE SPECIFICATION
INTERFACE MODEL
GESTURE-BASED INTERACTION MODEL
PLATFORM-SPECIFIC GESTURE SPECIFICATION
IMPLEMENT INTERFACE
A4. GENERATE GESTURE-BASED INTERFACE
A5. TEST GESTURES
INFORMATION SYSTEM INTERFACE
GESTURE-BASED INTERFACE
GESTURE RECOGNITION TOOL
ACTIVITY
PRODUCT
MULTI-PLATFORM CHOICES
INPUT / OUTPUT RELATIONSHIP
AUTOMATION
OUR PROPOSAL
HOST METHOD OR EXISTING PRODUCT

Fig. 1. gestUI method overview

The **computation-independent layer** is omitted because gestUI already assumes that the IS is going to be computerised. Note that gestUI is expected to be integrated into a full interface development method (represented with generic activities and artefacts in grey). Such a method can either be model-driven or code-centric. gestUI is user-driven because users participate in all non-automated activities; and it is iterative because it intends to discover the necessary gestures incrementally and provides several loopbacks. In the **platform-independent layer**, the gestures are defined (activity A1 in Fig. 1) by the developer but, preferably, in collaboration with representative users of the IS. Gestures are defined by sketching and are stored in the ‘Gesture catalogue model’, and is part of a larger ‘Interaction requirements’ specification. In the **platform specific layer**, a concrete gesture-recognition platform is selected (we currently support three platforms: quill [12], $N [8] and iGesture [13]). The ‘Platform-specific gesture specification’ (PSGS) is a machine-readable file format that can be interpreted by a gesture recognition tool. This specification can be automatically generated from the ‘Gesture catalogue model’ (during A3). The interface is also designed in this layer, so now the gesture-based interaction can also be determined (A2) in collaboration with the user. This mainly consists of defining correspondences between gestures and interface actions. In the **code layer**, the developer and the user can test the gestures using the gesture recognition tool (A5). The 'Gesture based interface' is automatically generated from the platform-specific layer artefacts (A4). The tool generates components (e.g. Java code) that are embedded into the IS interface.

4 The gestUI tool

The gestUI tool is developed using Java and Eclipse Modelling Framework. As shown in Fig. 2, the tool is structured in three modules. The numbers in brackets indicate the method activity each component supports. The method's internal products are not shown. The relationship with the external gesture recogniser is represented.



<<system>> gestUI TOOL
<<subsystem>> GESTURE CATALOGUE DEFINITION MODULE (A1)
<<subsystem>> GESTURE-ACTION CORRESPONDENCE DEFINITION MODULE
<<component>> GESTURE-BASED INTERACTION DESIGNER (A2)
<<component>> GESTURE-BASED INTERFACE GENERATOR (A4)
<<subsystem>> MODEL TRANSFORMATION MODULE (A3)
<<component>> $N M2T TRANSFORMATION
<<component>> quill M2T TRANSFORMATION
<<component>> iGesture M2T TRANSFORMATION
<<system>> INFORMATION SYSTEM
<<subsystem>> IS INTERFACE
<<component>> GESTURE-BASED INTERFACE
<<system>> $N, quill OR iGesture (A5)
GESTURES
INTER-ACTION
DEVELOPER AND USER
TARGET TECHNOLOGY
LEGEND:
COMPONENT
INFORMATION PROVISION
GENERATION
gestUI COMPONENT
EXTERNAL TECHNOLOGY
ROLE

Fig. 2. gestUI tool overview

4.1 Gesture catalogue definition module

It supports the definition of new multi-stroke gestures by means of an interface implemented in Java in which the user sketches the gestures. The set of gestures sketched by the user constitutes the 'Gesture catalogue model', conforms to the metamodel defined in this work (*Fig. 3*).

4.2 Model transformation module

It requires as data: the 'Gesture catalogue model', the target technology specified by the developer, and the target folder to save the output. Depending on the target technology, a different M2T transformation is executed which creates the PSGS, in the corresponding file format (i.e. XML for $N and iGesture and GDT 2.0 for quill). The transformation rules are written in Acceleo. The PSGS can be imported in a third-party gesture recogniser to test the gestures.

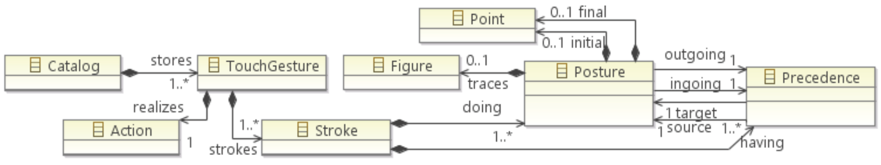


Fig. 3. Metamodel of the gesture catalogue modelling language

4.3 Gesture-action correspondence definition module

It allows the developer and the user to specify what action to execute whenever the gesture-based IS interface recognises a gesture. In a model-based IS interface development, the actions are specified in the interface model. In a code-centric interface development, they are implemented in the interface itself. We currently provide automated support to code-centric developments made in Java; that is, the gestUI module parses the source code of the user interface to obtain a list of actions. This module therefore requires two inputs: the previously created ‘Gesture catalogue model’ and the user interface (e.g. a Java code). The output of this module is the ‘Gesture-based interaction model’ and the same source code but now supporting the gesture-based interaction.

When generating the user interface Java source code, many references are included (e.g., to libraries to manage gestures, to libraries of the gesture-recognition technology (e.g. $N)), and some methods are added (e.g., to execute the gesture-action correspondence, and to capture gestures). Additionally, the class definition is changed to include some listeners, then the source code should obviously be compiled.

5 Demonstration of the method and tool

We demonstrate the integration of gestUI within a code-centric interface development method. For illustration purposes, we use a fictional, simple university management case and we narrate the project as if it actually happened. Fig. 4 shows the domain class diagram of a university with several departments, to which teachers are assigned and which manage classrooms. For the sake of brevity, we will just consider the two screens; namely, the initial and department management screens.

In the first method iteration, the university representatives tell the developer that they would like the gestures to resemble parts of the university logo.

Thus, they use the Gesture catalogue definition module to create a first version of the 'Gesture catalogue model' containing these three gestures: △ for departments, || for teachers and ☐ for classrooms. However, when the first interface design is available (see sketch in Fig. 5), they soon realise that other gestures are needed. This way, by defining new gestures, and after testing them, they determine that navigation will be done by means of the above-mentioned gestures, but that similar actions that appear across different screens shall have the same gestures (e.g. the gesture + shall be used to create both new departments and teachers).

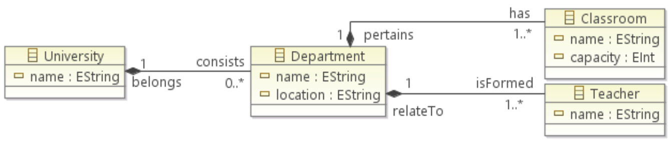
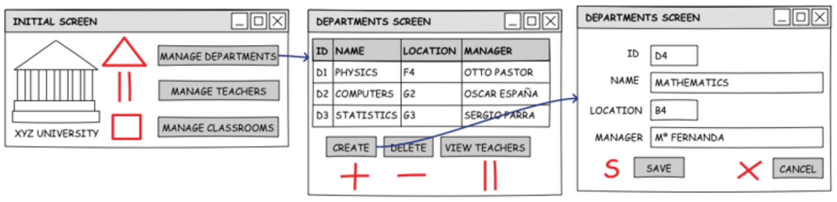


Fig. 4. UML class diagram of the demonstration case

The Model transformation module allows generating the PSGS for any of the available gesture-based recognition technologies (i.e. $N, quill and iGesture). The developer only needs to choose a single technology but we chose to demonstrate the multiplatform features of the gestUI method by generating the three gesture files. Using the appropriate tool, the users can test the gestures. Fig. 6 shows the gestures being recognised by the SN, quill and iGesture tools so the gestures have been properly converted by the Model transformation module.



ID	NAME	LOCATION	MANAGER
D1	PHYSICS	F4	OTTO PASTOR
D2	COMPUTERS	G2	OSCAR ESPAÑA
D3	STATISTICS	G3	SERGIO PARRA

ID D4
NAME MATHEMATICS
LOCATION B4
MANAGER Mª FERNANDA

Fig. 5. Screen mockups (gestures are shown in red, next to action buttons)

The developer assigns the gesture-action correspondence in collaboration with the user, supported by the Gesture-action correspondence definition module. The correspondences are informally shown in Fig. 5, next to each action button. Once the Java source code of the traditional interface is available, then the components that support the gesture-based interaction are generated. In this case, the chosen underlying gesture-recognition technology is $N; the users felt more comfortable with multi-stroke gestures (especially with regards to tracing some letters and symbols) so quill was discarded. The final IS interface consists of several screens that allow managing university information. Users can still interact with the IS in the traditional way (i.e. using the mouse), but now, they can also draw the gestures with a finger on the touch-based screen in order to execute the actions. Fig. 7 represents a specific interaction with the IS interface in which a department is being created.

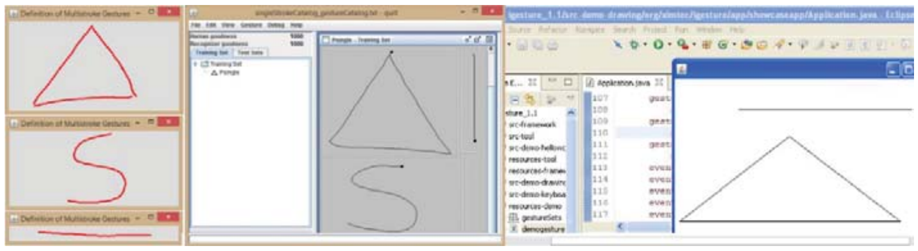
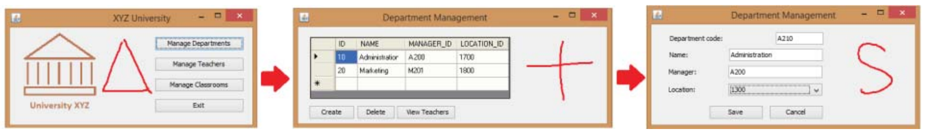

Definition of Multistroke Gestures
Definition of Multistroke Gestures
Definition of Multistroke Gestures

Fig. 6. An excerpt of multi-stroke gestures: $N (left) and quill (center) and iGesture (right)



ID	NAME	MANAGER_ID	LOCATION_ID
10	Administratior	A200	1700
20	Marketing	M201	1800

Department code: A210
Name: Administration
Manager: A200
Location: 1300

Fig. 7. Using gestures to execute actions on the interfaces

6 Summary and Future Work

We describe gestUI, a model-driven method, and the tool that supports it to specify multi-stroke gestures and automatically generating the information system components that support the gesture-based interaction. We validated the method and tool by applying them to a case and generated the Platform-specific gesture specification for three gesture-recognition technologies, to illustrate the multiplatform capability of the tool. The gestures were successfully recognised by the corresponding tools. We then automatically generated the final gesture-based interface components and integrated them into the IS interface. The advantages of the proposal are: platform independence enabled by the MDD paradigm, the convenience of including user-defined symbols and its iterative and user driven approach. Its main current limitations are related to the target interface technologies (currently, only Java) and the fact that multi-finger gestures are not supported. These limitations will be addressed in future work. We also plan further validation by applying the approach to the development of a real IS and to extending a CASE tool with gesture-based interaction (the Capability Development Tool being developed in the FP7 CaaS project). We also plan to integrate gestUI into a full-fledged model-driven framework capable of automatically generating the presentation layer, in order to extend it with gesture-based interaction modelling and code generation.

Acknowledgements

The author is grateful to his supervisors Sergio España and Óscar Pastor for their invaluable support and advice. This work has been supported by SENESCYT and Univ. de Cuenca - Ecuador, and received financial support from Generalitat Valenciana under Project IDEO (PROMETEOII/2014/039).

References

[1] D. Wigdor and D. Wixon, Brave NUI world: designing natural user interfaces for touch and gesture, USA: Morgan Kaufmann Publishers Inc., 2011.

[2] Fujitsu, "Touch- and gesture-based input to support field work," Fujitsu Laboratories Ltd., 18 02 2014. [Online]. Available: http://phys.org/news/2014-02-touch-gesture-based-field.html. [Accessed 24 11 2014].

[3] M. Hesenius, T. Griebe, S. Gries and V. Gruhn, "Automating UI Tests for Mobile Applications with Formal Gesture Descriptions," *Proc. of 16th Conf. on Human-computer interaction with mobile devices & services*, pp. 213-222, 2014.

[4] S. H. Khandkar, S. M. Sohan, J. Sillito and F. Maurer, "Tool support for testing complex multi-touch gestures," in *ACM International Conference on Interactive Tabletops and Surfaces, ITS'10*, NY, USA, 2010.

[5] L. Spano, A. Cisternino and F. Paternò, "A Compositional Model for Gesture Definition," *LNCS Human-Centered Soft. Eng.*, vol. 7623, pp. 34-52, 2012.

[6] K. Kin, B. Hartmann, T. DeRose and M. Agrawala, "Proton++: A Customizable Declarative Multitouch Framework," in *UIST'12*, Cambridge, USA, 2012.

[7] Ideum, "GestureML," Ideum, 22 November 2014. [Online]. Available: http://www.gestureml.org/. [Accessed 6 December 2014].

[8] L. Anthony and J. O. Wobbrock, "A Lightweight Multistroke Recognizer for User Interface Prototypes," *Proc. of Graphics Interface*, pp. 245-252, 2010.

[9] M. Karam and M. C. Schraefel, "A taxonomy of Gestures in Human-Computer Interaction," in *Retrieved from http://eprints.soton.ac.uk/261149/*, 2005.

[10] M. Guimaraes, V. Farinazzo and J. Ferreira, "A Software Development Process Model for Gesture-Based Interface," in *IEEE International Conference on Systems, Man, and Cybernetics* , Seoul, Korea, 2012.

[11] M. Nielsen, M. Storring, T. Moeslund and E. Granum, "A Procedure for Developing Intuitive and Ergonomic Gesture Interfaces for Man-Machine Interaction," Aalborg University, Aalborg, Denmark, 2003.

[12] A. C. Long and J. Landay, Quill: a gesture design tool for pen-based user interfaces, Berkeley: University of California, 2001.

[13] B. Signer, M. Norrie and U. Kurmann, "iGesture: A General Gesture Recognition Framework," in *Proceedings of ICDAR 2007, 9th Int. Conference on Document Analysis and Recognition*, Brazil, 2007.

A Process Mining Technique Using Pattern Recognition

Veronica Liesaputra[1], Sira Yongchareon[1], and Sivadon Chaisiri[2]

[1]Department of Computing and Information Technology
Unitec Institute of Technology, New Zealand
vliesaputra@unitec.ac.nz, sira@maxsira.com

[2]School of Information Technology
Shinawatra University, Thailand
sivadon@ieee.org

Abstract. Several works have proposed process mining techniques to discover process models from event logs. With the existing works, mined models can be built based on analyzing the relationship between any two events seen in event logs. Being restricted by that, they can only handle special cases of routing constructs and often produce unsound models that do not cover all of the traces in the logs. In this paper, we propose a novel technique for process mining based on using a pattern recognition technique called *Maximal Pattern Mining* (MPM). Our MPM technique can handle loops (of any length), duplicate tasks, non-free choice constructs, and long distance dependencies. Furthermore, by using the MPM, the discovered models are generally much easier to understand.

1. Introduction

Since the mid-nineties, several techniques have been proposed to automatically discover process models from event logs in both software processes and business process domains [2, 3, 4]. Several algorithms are variants of the α-algorithm (e.g., in [5, 6, 7, 8]), which is regarded as a well-known technique for process discovery that pioneered studies in this field. Nevertheless, due to the fact that the α-algorithms face problems dealing with complicated routing constructs, noise, and incompletes [1], more advanced techniques, such as region-based approaches (e.g., [14, 15, 16, 18]), heuristic mining [9], fuzzy mining [10], and genetic mining [11], have been proposed to tackle these aforementioned problems.

We argue that the existing algorithms for discovering process models are still unable to efficiently and accurately handle loops (of any length), duplicate tasks, concurrency, long dependencies and complex routing constructs. In fact, some of these algorithms may produce unsound models. To address these problems, we propose a novel process mining technique called *Maximal Pattern Mining* (MPM). Instead of mining the relationship between two events, MPM mines a set of patterns that could cover all of the traces seen in an event log. The time needed by our algorithm to process mine and generate a process model is also significantly shorter than all the existing algorithms.

The remainder of the paper is organized as follows. Section 2 reviews and discusses the work that has been done in the process mining area. Section 3 proposes our MPM technique for process discovery. Section 4 discusses our preliminary evaluation. Finally, the conclusion and future works are given in Section 5.

2. Background and Related Work

Van der Aalst et al. [5] proposed α-algorithm to discover structured workflow nets from complete event logs. However, the α-algorithm cannot cope with noise, incompleteness of workflow logs, short loops, and non-free choice constructs. Later, Alves de Medeiros et al. [6] developed α^{+}-algorithm, an improved version of α-algorithm, which is capable of detecting short loops. Further, Wen et al. [7, 8] proposed α^{++}-algorithm to discover non-free choice constructs and β-algorithm to detect concurrency. Due to the fact that all α-algorithms face the same robustness problem, Weijters et al. [9] proposed Heuristics Miner by extending the α-algorithm to analyze the frequency of the three types of relationships between activities in a workflow log: direct dependency, concurrency, and not-directly connectedness. In contrast to the α-algorithms, Gunther and van der Aalst [10] proposed Fuzzy Miner, an adaptive technique to discover behavior models from an event log using significance and correlation measures.

Van der Werf et al. [16] proposed a discovery technique using Integer Linear Programming (ILP) based on the theory of regions. Van der Aalst et al. [14] proposed a Finite State Machine (FSM) Miner/Petrify two-step approach to find a balanced trade-off between generalization and precision of discovered process models. The theory of region is used in their approach as a method to bridge FSM and Petri-Net models as also proposed in [15]. Sole and Carmona [18] presented an aggressive folding region-based technique, which is based on the theory of region, to reduce the total number of states of a transition system and speed up the discovery process. Alves de Medeiros et al. [11] proposed a genetic algorithm which performs a global search based on the use of fitness function using both a recall and a precision measure to find the best matched models. The DT Genetic and Genetic Miner can detect non-local patterns and, due to its post-pruning step, it has a reasonable robustness. While the latter cannot detect duplicate tasks, the former can detect them. Similarly, Goedertier et al. [12] proposed AGNEsMiner to deal with problems such as expressiveness, noise, incomplete event logs, and the inclusion of prior knowledge by representing process discovery as a multi-relational classification problem [13] on event logs supplemented with Artificially Generated Negative Events (AGNEs). This technique can learn the conditions that distinguish between the occurrence of either a positive or a negative event.

Based on the above discussion, we have observed that only the DT Genetic Miner [11] can tackle all of the typical process mining problems, i.e., noise, duplicate tasks, hidden tasks, non-free choice constructs, and loops. However, because of the nature of the genetic algorithm, it consumes much more processing time and space in order to learn and construct a model. Mining efficiency is considered a major drawback of this approach in which it is undesirable, especially when it is applied to a complicated real-life log. To overcome such issues, we need to develop a better technique that not only solves all the typical process mining problems but also requires far less processing time.

3. Maximal Pattern Mining (MPM)

Instead of looking at the relationship between two events which is what most of the existing process mining techniques focus on, we propose a pattern mining technique to analyse the whole sequence of events in all of the traces and find the optimal set of “regular expression”-like patterns that would cover them. Our MPM technique is described in Sections 3.1. Assumptions and limitations of the technique are discussed in Section 3.2.

3.1. Overview

Let $T = \{t_0, t_1 \ldots t_n\}$ be the collections of all the traces in an event log that is ordered first by the value of the events in the trace and then by the number of events in the trace. A trace t_n is an ordered sequence of events or completed tasks, $t_n = \langle z_0, z_1 \ldots z_m \rangle$. We denote $|t_n|$ as the number of events in a trace. An event z_m only contains 1 event type, i.e. $|z_m| = 1$. All the traces and events in T and t_n are not unique, i.e. it is possible to have $T = \{\langle a,b,c,b,b,c,d,e\rangle,\langle a,b,c,b,b,c,d,e\rangle,\langle a,b,b,c,e,d\rangle\}$. Given an input T, our algorithm will first create a list of unique patterns $P = \{p_0, p_1 \ldots p_i\}$ and then generate a graph based on P. The following sections will describe each of them. A pattern $p_i = \langle e_0, e_1 \ldots e_j\rangle$ is an ordered sequence of elements, $|p_i|$ is the number of elements in the pattern and $p_i.support$ is the number of traces covered by the pattern. An element $e_j = \{v_0, v_1 \ldots v_k\}$ contains k number of unique event types (i.e. $|e_j| = k$) and $e_j.loop$ is a list of $\langle v_k\colon w\rangle$ tuples that indicate whether v_k is self-looping ($w = \{v_k\}$) and/or is the last element of a sequence-loop ($w = \{e_x\ e_{x+1} \ldots e_{x+y}\}$ and $e_{x+y} = v_k$). The *loop* list is ordered first by the event value and then by the number of elements in w ($|w|$). An element’s value v_k only contains 1 event type. All the elements inside p_i might not be unique. For instance, given the $T = \{\langle a,b,c,b,b,c,d,e\rangle, \langle a,b,c,b,b,c,d,e\rangle, \langle a,b,b,c,e,d\rangle\}$ specified above, our algorithm will only produce 1 pattern in P. $p_0 = \langle e_0, e_1, e_2, e_3\rangle$, where $e_0 = a$ and $e_0.loop = \emptyset$; $e_1 = b$ and $e_1.loop = \emptyset$; $e_2 = c$ and $e_2.loop = \{\langle c\colon \{bc\}\rangle\}$; and $e_3 = \{d, e\}$ and $e_3.loop = \emptyset$. Elements with more than one event type indicate a parallelization. In our example, e_3 shows that in the last 2 events of our model the values could be either *de* or *ed*. Because p_0 covers all the traces in T, $p_0.support = 3$.

Our graph algorithm will then generate the following model (Fig. 1) based on p_0. We use the operator AND to indicate the set of tasks that are running at the same time, and XOR to indicate a path selection.

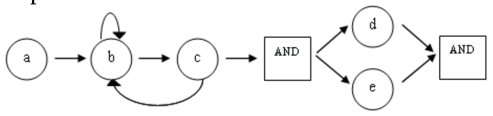


Figure 1. The generated model for $\{\langle a,b,c,b,b,c,d,e\rangle,\langle a,b,c,b,b,c,d,e\rangle,\langle a,b,b,c,e,d\rangle\}$

The algorithm we use to construct the most optimal patterns for a given trace of events has five main phases: finding self and/or sequence loops, storing the pattern in a vertical format, identifying events that should be done concurrently, investigating whether a trace is covered by a pattern in P, and pruning non-maximal patterns.

Loops. A sequence of elements $S = \langle s_0, s_1 \ldots s_q\rangle$ is in a loop in the trace $t_n = \langle z_0, z_1 \ldots z_m\rangle$ or in the pattern $p_i = \langle e_0, e_1 \ldots e_m\rangle$ if and only if there is a sequence of elements such that for all $b \in \{0\ldots q\}$and $q \le (m - a)/2$, $z_{a+b} = s_b$ and $z_{a+q+b} = s_b$ or $e_{a+b} = s_b$ and $e_{a+q+b} = s_b$, where a is the starting index where S occurs in the trace or in the pattern ($0 \le a \le m$). The first phase of our pattern mining is to identify these loops. For every $S+$ occurring in t_n and p_i, we replace it with S and set the *loop* property of the last element in S. For instance, given a pattern *⟨a,b,b,c,d,{e,f},c,d,{e,f}c,d,{e,f}g⟩*, the pattern becomes *⟨a,b,c,d,{e,f}g⟩* where the loop property for *b* is *b*, and the loop property for *{e,f}* is *cd{e,f}*. By identifying loops first, MPM would be able to deduce that traces *⟨a,b,d,d,c,b,b,b,d,c,b,d,c,e⟩* and *⟨a,b,d,c,b,d,d,c,e⟩* are the same and are both covered by the pattern *⟨a,b,d,c,e⟩*.

Vertical Representation. Existing process mining algorithms require several scans of the event logs or need to maintain large amounts of intermediate candidates in the main memory to generate process models [7, 8, 11, 13]. To alleviate this problem, MPM stores all patterns in the vertical format as an IdList in bitset representation [20] where each entry represents an element with the id of the trace where the element appears (*id*) and the position (*pos*) where it appears. The support of a pattern is calculated by making joint operations with IdLists of smaller patterns. Thus, MPM would only need to perform a single scan through the log to generate an IdList of patterns containing single elements (see [20] for details). To make it more verbose, MPM uses the symbol $ to indicate the end of a trace. Given $T = \{\langle a,b,c,b,b,c,d,e,a\rangle,\langle a,b,b,c,e,d,a\rangle,\langle e,d,a\rangle\}$, the vertical representation (V_T) of it is represented as follows:

A	
id	**pos**
0	0, 5
1	0, 5
2	2

b	
id	**pos**
0	1
1	1

c	
id	**pos**
0	2
1	2

d	
id	**pos**
0	3
1	4
2	1

e	
id	**pos**
0	4
1	3
2	0

$	
id	**pos**
0	6
1	6
2	3

Concurrency. The next phase of our pattern mining is to identify tasks that should be done in parallel. A set of events $V = \{v_0, v_1 \ldots v_q\}$ are performed at the same time if and only if there are at least q number of unique traces with the following sequence $\langle z_0, z_1 \ldots z_{a-1}\ z_a, z_{a+1} \ldots z_{a+q}\ z_{a+q+1}, z_{a+q+2} \ldots z_m\rangle$, where the sequence $\langle z_0, z_1 \ldots z_{a-1}\rangle$ and $\langle z_{a+q+1}, z_{a+q+2} \ldots z_m\rangle$ have the same pattern across those traces, there are no events mentioned more than once in $\langle z_a, z_{a+1} \ldots z_{a+q}\rangle$, and for all $b \in \{0\ldots q\}$and $q \le (m - a)$, $z_{a+b} \subseteq V$, where a is the starting index where a combination of all the events in V occur ($0 \le a \le m$). Sequence $\langle z_0, z_1 \ldots z_{a-1}\rangle$ and $\langle z_{a+q+1}, z_{a+q+2} \ldots z_m\rangle$ may be $\emptyset$. Instead of $z_{a+b} = V$, we relax the criteria to $z_{a+b} \subseteq V$ with the assumption that if we see almost all of V possible events combined in T, it must be that the trace log is incomplete. For example, given a set of traces *{⟨a,b,c,d,e⟩,⟨a,b,d,c,e⟩,⟨a,c,d,b,e⟩}*, we first look at the first two traces where we get *⟨a,b,{c,d},e⟩* as it is possible to switch the position of task *c* and *d* around. We then compare it with the last trace where we get *⟨a,{b,c,d},e⟩* as we can switch the position of task *c* and *d* around with *b*. In the future, we may use the trace frequency to help us decide when we should use the strict or relaxed criteria.

Coverage. A pattern $p_i = \langle e_0, e_1 \ldots e_n\rangle$ specifies the sequence of patterns that covers some of the traces in T and it can be represented as a deterministic finite automata DFA_i with (a) a well-defined start state, (b) one or more accepted states and (c) deterministic transitions across states on symbols of the event values. A trace $t_n = \langle z_0, z_1 \ldots z_m\rangle$ is covered by the pattern p_i if and only if the sequence of transitions for the elements of t_n

from the start state results in an accepted state. Fig. 2 illustrates the deterministic finite automaton for the pattern ⟨*a,b,{c,d},e*⟩ with the *loop* property for *b* to be *b*. We use > to indicate the start state and double circles for the accept state. The diagram shows that the pattern covers the following set of traces *{⟨a, b, c, d, e⟩, ⟨a, b, d, c, e⟩, ⟨a, b, b, c, d, e⟩, ⟨a, b,…,b, c, d, e⟩, ⟨a, b,…,b, d, c, e⟩}*. However, it will reject the following set of traces *{⟨a,b⟩, ⟨e⟩, ⟨a,b,h⟩, ⟨a,b,c,d⟩, ⟨a,b,b,d,c⟩, ⟨a,b,a,d,c,e⟩}*. Because we have to go through each of the elements of t_n to identify events that should be done in parallel, we perform both tasks simultaneously.

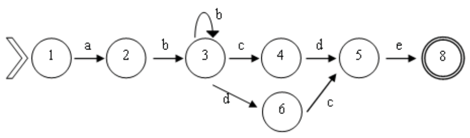


Figure 2. The deterministic finite automata model for p_i = ⟨*a,b,{c,d},e*⟩

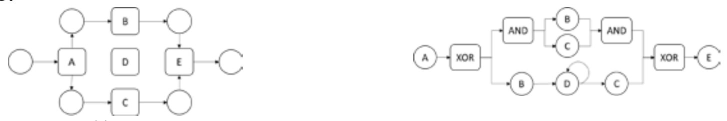
Maximal Patterns. A pattern p_i is said to be maximal if and only if there is no other pattern p_j in *P* that has the same start and accept states and covers the same or more traces in *T*. Given *P* = *{⟨a,b,c,d⟩,⟨a,{b,c},d⟩,⟨a,b,c⟩}*, only p_1 and p_2 are maximal because p_0 is a sub-pattern of p_1.
Noise. To further filter *P* from noisy data, we set a support threshold value, *thresh*, such that we would only keep frequent patterns p_i and events v_k, i.e. $p_i.support \geq thresh$ and $v_k.support \geq thresh$. All patterns and events are accepted if the threshold value is 0.

3.2. Assumptions and Limitations

An event in a transactional log usually contains information such as the event type/value (e.g. apply for a drivers licence or update a patients information), the agent/performer that initiates the event, timestamp and the data element being modified or accessed (e.g. the age of a patient, the driving test result). Because the goal of MPM is to find all possible orderings of the logged events in the system, only the event's type or value are mined. Other information, such as the timestamp and agent, are removed from the logs. In our setting, we know the original model that our algorithm should strive to construct, the complete list of traces that the model can generate, and the instances in a log that are negative examples. But in real life scenarios, no original model is available. Logs may contain noise such as mislabelled events, incorrectly logged sequences of events and exceptions. In fact, a particular trace of events observed does not mean that the model should be able to reproduce it. Furthermore, in a complex process with many possible paths, only a fraction of those paths may be present in the log, i.e., the log is incomplete. Thus, it is undesirable to construct a model that allows only for the observed instances in the log. Since we do not know which instance in the log is noise, we assume that every trace/event that is recorded in the log and appears no less than a user's specified threshold frequency is correct (positive examples). However, unobserved traces of events are not considered as negative examples. Our MPM algorithm can construct a model that can explain all the traces of events found in the logs while also allowing for any unobserved behaviour.

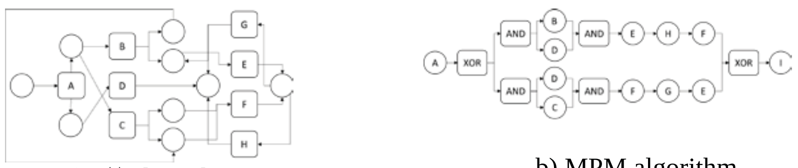
4. Preliminary evaluation

We evaluated the quality of the mined model produced by MPM, α^{++}, DT genetic miner, AGNEs and heuristic miners according to logs that are mentioned in their respective publications. We did not perform the evaluation on α and α^{+} as [7, 8] have reported that α^{++} can construct a model that handles more complex control-flow constructs. Similar to other discovery algorithms, our MPM algorithm is implemented as a plugin of ProM [19]. In our initial evaluation, we use synthetic log data to demonstrate the fact that the MPM algorithm can significantly improve the performance of the existing approaches, especially the α-algorithm and its variants. We do not use parameter fine-tuning or metadata to enhance the performance of our algorithm. We have also used the default settings for α^{++}, genetic miner and AGNEs. To further extend the capability of Heuristic Miners, we configure it to discover long distance dependencies based on completed events' values and positions on a trace. Due to the fact that the α^{++} algorithm builds a process model based on the relationship between *any two* events so that it does not allow an event to occur more than once in the model, it requires additional heuristics to handle long distance dependency, short loops (maximum of two events) and non-free-choice constructs (combination of choice and concurrency); and assumes that two or more events must occur concurrently if they have the same parents (i.e. low precision). Therefore, it is possible for the α^{++} algorithm to produce *unsound* workflow nets as shown Figures 3 and 4. Similarly, because Heuristic Miners also builds a casual matrix that represents the relationship between any two events, it cannot handle duplicate tasks as illustrated in Figure 5. Although AGNEs is more versatile than Heuristic Miners, it is still incapable of handling a complex non-free choice construct such as is displayed in Figure 6.



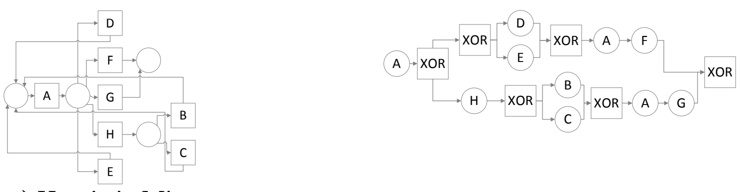
a) α^{++} algorithm b) MPM algorithm

Figure 3. Log *T* = {*ABCE, ACBE, ABDDCE*}



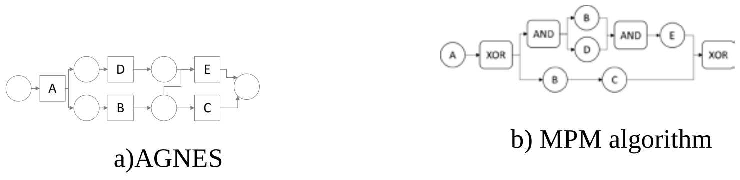
a) α^{++} algorithm b) MPM algorithm

Figure 4. Log *T* = {*ABDEHFI, ADBEHFI, ACDFGEI, ADCFGEI*}



a) Heuristic Miner b) MPM algorithm

Figure 5. Log *T* = {*ADAF, AEAF, AHBAG, AHCAG*}

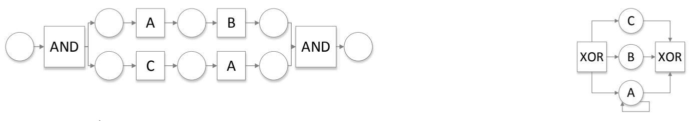


b) MPM algorithm

a)AGNES

Figure 6. Log *T* = {*ABC, ABDE, ADBE*}

Our MPM algorithm discovers a process model by reading patterns from the *whole sequence* of events in the traces. Thus, its criteria is more stringent than Heuristic Miners or α^{++}; it can handle duplicate tasks, long distance dependencies, loops of any length and non-free choice constructs. The process model discovered by MPM is always sound, and it is generally more accurate and readable than the models mined by AGNEs, Heuristic Miners or α^{++}. However, MPM is incapable of generating a model that accurately represents duplicate tasks in a parallel process structure, as shown in Figure 7. DT Genetic Algorithm is the only algorithm that can correctly mine this log.



a) DT Genetic Miner

b) MPM algorithm

Figure 7. Log *T* = {*ABC, ABDE, ADBE*}

While DT Genetic Miner will sometimes produce a model that is more accurate than MPM, MPM can generate a similar model in significantly less time. Furthermore, MPM can build and improve the mined model incrementally in near real time as it receives new traces of events, i.e. the model becomes more accurate as it sees more unique traces of events.

5. Conclusion and Future work

In this paper, we propose a novel technique called *Maximum Pattern Mining* (MPM) to discover a process model from event logs. We have implemented our technique with preliminary evaluations against well-known process discovery algorithms: α^{++}, DT genetic miner, AGNEs and the Heuristic Miners algorithm. Our results show that it can handle more general cases, such as loops of any length and long distance dependencies. In the future, we will implement and improve this technique with evaluations on real-life logs to see if our algorithm can handle very complex and very large logs.

References

1. van der Aalst, W.M.P.: Process Mining: Overview and Opportunities, *ACM Transactions on Management Information Systems*, 2012, vol. 3, no. 2, article 7.

2. Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs, in: *Proceedings of the 6th International Conference on Extending Database Technology* (EDBT'98), 1998, LNCS 1377, pp. 469-483.
3. Cook, J., Wolf, A.: Discovering models of software processes from event-based data, *ACM Transactions on Software Engineering and Methodology,* 1998 (7), pp. 215–249.
4. Datta, A.: Automating the discovery of AS-IS business process models: probabilistic and algorithmic approaches, *Information Systems Research*, 1998, vol. 9, pp. 275–301.
5. 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*, 2004, vol. 16, pp. 1128–1142.
6. Alves de Medeiros, A.K., van Dongen, B.F., van der Aalst, W.M.P., Weijters, A.J.M.M.: Process Mining: Extending the Alpha-Algorithm to Mine Short Loops, *BETA Working Paper Series*, TU Eindho- ven, 2004, vol. 113.
7. Wen, L., van der Aalst, W.M.P., Wang, J., Sun, J.: Mining process models with non-free-choice constructs, *Data Mining and Knowledge Discovery,* 2007 (15), pp. 145–180.
8. Wen, L., Wang, J., van der Aalst, W.M.P., Huang, B., Sun, J.: A novel approach for process mining based on event types, *Journal of Intelligent Information Systems*, 2009, vol. 32, pp. 163–190.
9. Weijters, A.J.M.M., van der Aalst, W.M.P., Alves de Medeiros, A.K.: Process Mining with the Heuristics Miner algorithm, *BETA Working Paper Series*, 2006, TU Eindhoven, vol. 166.
10. Günther, C.W., van der Aalst, W.M.P.: Fuzzy mining - adaptive process simplification based on multi-perspective metrics, in: *Proceedings of the 5th International Conference on Business Process Management* (BPM), 2007, LNCS 4714, pp. 328–343.
11. Alves de Medeiros, A.K., Weijters, A.J.M.M., van der Aalst, W.M.P.: Genetic process mining: an experimental evaluation, *Data Mining and Knowledge Discovery,* 2007, vol. 14, pp. 245–304.
12. Goedertier, S., Martens, D., Vanthienen, J., Baesens, B.: Robust process discovery with artificial negative events, *Journal of Machine Learning Research*, 2009 (10), pp. 1305–1340.
13. Blockeel, H., De Raedt, L.: Top-down induction of first-order logical decision trees, *Artificial Intelligence*, 1998, vol. 101, pp. 285–297.
14. van der Aalst, W.M.P., Rubin, V., Verbeek, H.M.W., van Dongen, B.F., Kindler, Günther, C.W.: Process mining: a two-step approach to balance between underfitting and overfitting, *Software and System Modeling*, 2010 (9), pp. 87–111.
15. Carmona, J., Cortadella, J., Kishinevsky, M.: New region-based algorithms for deriving bounded Petri nets, *IEEE Transactions on Computers*, 2010 (59), pp. 371–384.
16. van der Werf, J.M.E.M., van Dongen, B.F., Hurkens, C.A.J., Serebrenik, A.: Process discovery using integer linear programming, *Fundamenta Informaticae*, 2009, vol. 94, pp. 387–412.
17. Ferreira, D.R., Gillblad, D.: Discovering process models from unlabelled event logs, in: *Proceedings of the 7th International Conference on Business Process Management* (BPM), 2009, LNCS 5701, pp. 143–158.
18. Sole, M., Carmona, J.: Region-Based Folding in Process Discovery, *IEEE Transactions on Knowledge and Data Engineering*, 2013, vol. 25(1), pp. 192-205.
19. Günther, C.W., Verbeek, E.: XES Standard version 2, 2014, http://www.xes-standard.org/_media/xes/xesstandarddefinition-2.0.pdf
20. Ayres, J., Flannick, J., Gehrke, J., Yiu, T.: Sequential pattern mining using a bitmap representation, in: Proc. 8th ACM Intern. Conf. Knowl. Discov. Data Mining, ACM (2002), pp. 429–435.

OBIS: Ontology-Based Information System Framework

Kārlis Čerāns[1], Aiga Romāne[1]

karlis.cerans@lumii.lv, aiga.romane@inbox.lv
Institute of Mathematics and Computer Science, University of Latvia
Raina blvd. 29, Riga, LV-1459, Latvia

Abstract. We demonstrate a framework for automated information system generation from a given data ontology describing either an existing data set, or a data set to be filled up and maintained using the created information system. The IS data are stored into real or virtual RDF data stores and are accessed by means of SPARQL queries. The data ontology structure information and annotations allow automated generation of IS reflecting the data structure specified in the ontology.

Keywords: Ontologies, RDF schemas, automated information system generation, user interface annotations.

1 Introduction

The semantic web standards, including RDF [1,2], OWL [3] and SPARQL [4] define information representation and querying infrastructure that is basis for Semantic Web [5] and Linked Data [6], as well as enterprise-level semantic technology use. The semantic technologies offer much higher-level view on data than do the classic relational databases (RDB) with their corresponding SQL query language thus raising a hope of more direct involvement of various domain experts in data set definition, exploration and analysis. There are both a W3C standard R2RML [7], as well as numerous developments (e.g. Virtuoso RDF Views [8], D2RQ [9], ontop [10] and RDB2OWL [11]) for mapping relational database information into the semantic technology landscape. The RDF data stores such as Virtuoso [8] and Stardog [12] provide native RDF data storage so facilitating the RDF data availability. For the semantic information landscape implementation there is also a need of tools allowing creation, access and analysis of the available data.

We demonstrate here a system for automated information system user interface creation from the structure of data described as annotated OWL ontology thus allowing browsing the available RDF data, as well as editing the data, if the corresponding RDF data store allows for data updates and data entry. There are two basic use cases of the OBIS Framework: (i) to browse and analyze data in existing

[1] Partially supported by Latvian State Research programme NexIT project No.1 'Technologies of ontologies, semantic web and security'

real or virtual RDF databases, including the RDF databases that are conceptual representations of relational database data [13,14], as well as arbitrary SPARQL endpoints that have a corresponding data schema description available, and (ii) to manage information system data primarily stored into a RDF database (including data inserting and updating); thus creating a SPARQL-enabled conceptual data store.

We structure the presentation into two main sections – first is explained the basic OBIS application generation, followed by application tuning possibility description. The conceptual foundation of the OBIS system has been described in [15], the current paper reports on new basic functionality (e.g. related items view), as well as entirely new ontology annotation framework.

2 OBIS: Basic Working Scheme

The ontology-based information system creation in OBIS starts with obtaining or designing of data ontology reflecting the structure of the data underlying the information system. Figure 1 contains an example mini-University ontology in the OWLGrEd[2] ontology editor notation [16] describing OWL classes as UML classes, OWL object properties as role links between property domain and range classes and OWL data properties as their domain class attributes.

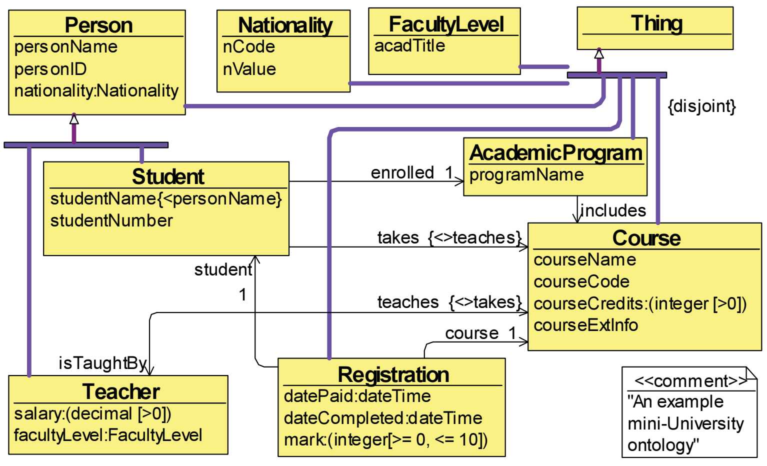
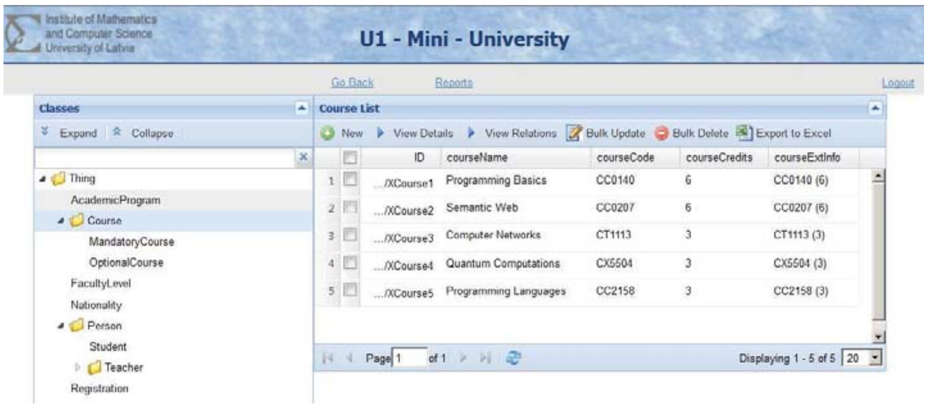


Fig. 1. Example: the OWLGrEd notation for a mini-university ontology

With the OBIS framework up and running on a web-server the user creates a new application in the OBIS Application Manager. The user then fills in the application name as well as ontology namespace (from the data ontology) and data namespace (not relevant for read-only ontology data access), chooses the repository type (e.g. Virtuoso server[3], a SPARQL-endpoint or a Jena TDB store) and connection

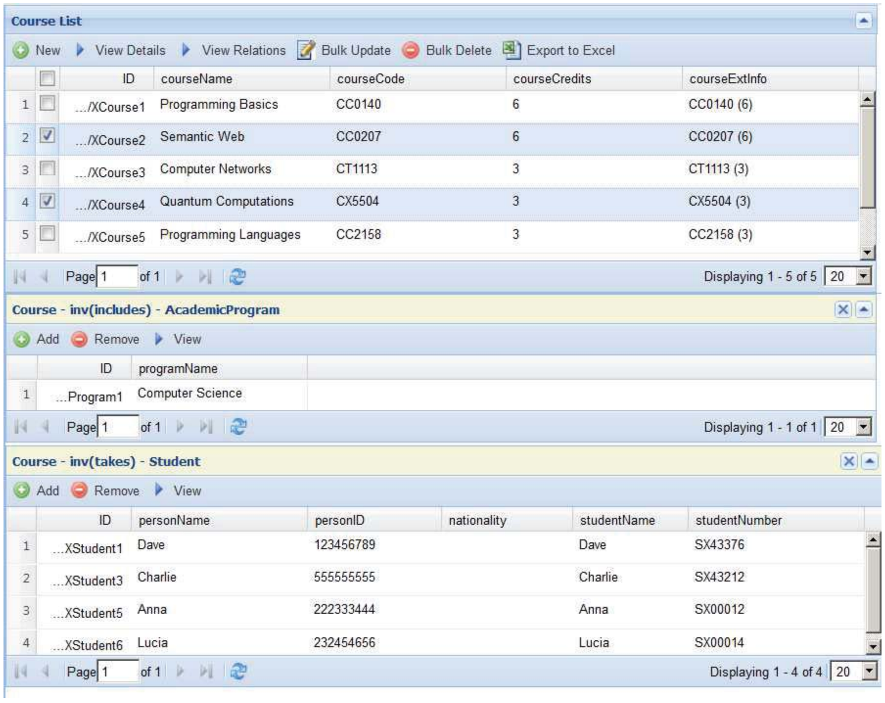
[2] The editor can be downloaded from http://owlgred.lumii.lv
[3] To be downloaded from e.g. http://virtuoso.openlinksw.com/ and installed separately

parameters. After saving the created basic application information and loading the data ontology the concrete application based on the data ontology can be generated and run. Figure 2 shows mini-University example case of the obtained application working interface, consisting of a class tree browser with the option to select a class and obtain a table view of its instance attributes.



	ID	courseName	courseCode	courseCredits	courseExtInfo
1	.../XCourse1	Programming Basics	CC0140	6	CC0140 (6)
2	.../XCourse2	Semantic Web	CC0207	6	CC0207 (6)
3	.../XCourse3	Computer Networks	CT1113	3	CT1113 (3)
4	.../XCourse4	Quantum Computations	CX5504	3	CX5504 (3)
5	.../XCourse5	Programming Languages	CC2158	3	CC2158 (3)

Fig. 2. Example: the generated mini-University information system: the instance table view for Course class.



		ID	courseName	courseCode	courseCredits	courseExtInfo
1	☐	.../XCourse1	Programming Basics	CC0140	6	CC0140 (6)
2	☑	.../XCourse2	Semantic Web	CC0207	6	CC0207 (6)
3	☐	.../XCourse3	Computer Networks	CT1113	3	CT1113 (3)
4	☑	.../XCourse4	Quantum Computations	CX5504	3	CX5504 (3)
5	☐	.../XCourse5	Programming Languages	CC2158	3	CC2158 (3)

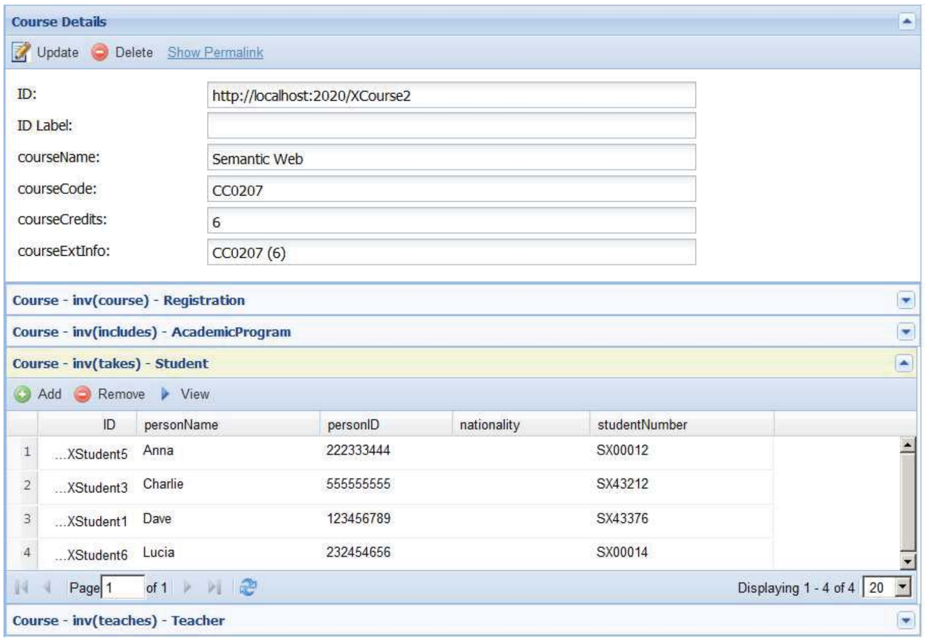
	ID	programName
1	...Program1	Computer Science

	ID	personName	personID	nationality	studentName	studentNumber
1	...XStudent1	Dave	123456789		Dave	SX43376
2	...XStudent3	Charlie	555555555		Charlie	SX43212
3	...XStudent5	Anna	222333444		Anna	SX00012
4	...XStudent6	Lucia	232454656		Lucia	SX00014

Fig. 3. The related item view options for the instance list in OBIS

Further ontology data exploration options include the sorting and filtering, export to MS Excel as well as related data item view allowing the user to select one or several instance links defined for the class in the ontology as well as a list of instances and obtain the lists of items connected to the selected ontology instances via the specified links. Figure 3 demonstrate this facility for the courses ‘Semantic Web’ and ‘Quantum Computations’ in the mini-University example (note the properties *includes* and *takes* coming into the *Course* class in the data ontology in Figure 1).

There is also a detailed view form for instances with fields for attribute values and links in the ontology data coming in (the direct links) and going out (the inverse link), as in Figure 4. Show Permalink option helps sharing the instance property page among various developers working with the same OBIS server application.


Course Details

Update Delete Show Permalink

ID:	http://localhost:2020/XCourse2
ID Label:	
courseName:	Semantic Web
courseCode:	CC0207
courseCredits:	6
courseExtInfo:	CC0207 (6)

Course - inv(course) - Registration

Course - inv(includes) - AcademicProgram

Course - inv(takes) - Student

Add Remove View

	ID	personName	personID	nationality	studentNumber
1	...XStudent5	Anna	222333444		SX00012
2	...XStudent3	Charlie	555555555		SX43212
3	...XStudent1	Dave	123456789		SX43376
4	...XStudent6	Lucia	232454656		SX00014

Page 1 of 1 Displaying 1 - 4 of 4 20

Course - inv(teaches) - Teacher

Fig. 4. A class instance view, with link options.

OBIS can be used not only as the ontology data information viewer. In the case, if the underlying RDF data store supports SPARQL data updates, there is possibility of new information entry, as well as information update and delete in OBIS, as well. The class instance information entry and edit forms are similar to instance view forms of Figure 4 (with information save option enabled); the link information is updated via the *Add* option in the tables showing link relations.

Further options in OBIS are report creation that allows also execution of textual SPARQL queries against the underlying RDF data store as well as form and report information export to MS Excel. The internal report definition tool in OBIS is work in progress. To ease writing textual SPARQL queries one can use e.g. a ViziQuer tool[4] [17], among the others.

3 Enhancing OBIS Applications

An information system user interface that is based just on the reproduction of the structure of ontology classes and data and object properties (with an analogy with RDB table, attribute and link structure) may appear to be too simplistic for a useful information system. One could desire e.g. more specific table and form view definitions for particular classes. In particular, the classes containing the classifier data, or, in general, the links among classes with their target maximum cardinality not exceeding 1 (i.e., the n:1 links) may benefit from special visual representation in the user interface. There are three basic mechanisms of enhancing OBIS applications:

- specification of cardinality information in the ontology (this requires using the standard means available in OWL, so the effects of cardinality information availability can be considered basic OBIS application generation aspect);
- extending OWL ontology with specific user interface annotations, and
- customizing (configuring) the application after its initial generation.

The main user interface annotations, supported by OBIS, are the following[5]:

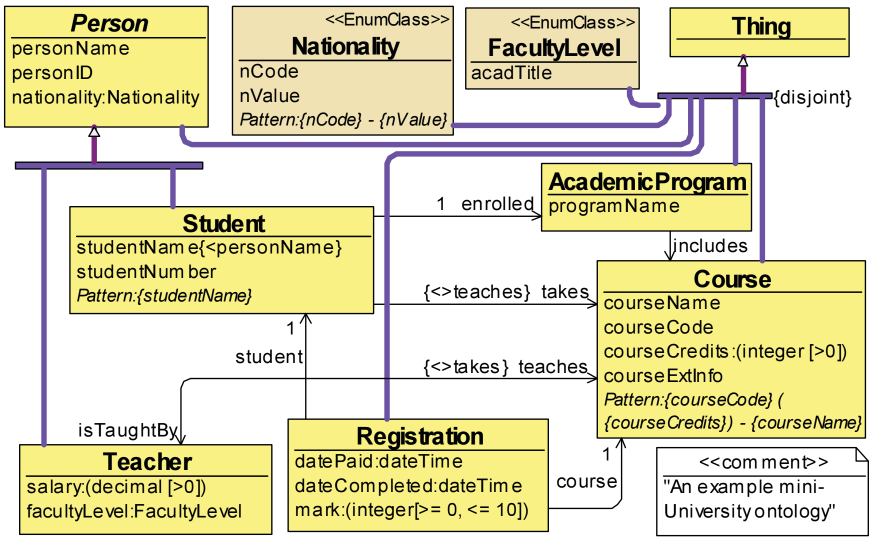
- *obis:textPattern.* Defines textual form for class instance presentation via relations with maximum cardinality 1. The textual form of linked instances of such classes are included into linking instance presentations both in table and form and edit views (a combo-box is used for the edit view).
- *obis:isEnumerated*. Specifies a class to be a classifier class (an enumerated class), so that a textual form of its instance presentation is available disregarding an explicit *obis:textPattern* specification by the user.
- *obis:defaultOrder*. Specifies the default order of attributes (both data attributes and classifier attributes) for class instances within a class table or form view. The OWL ontology axiom structure does not foresee a possibility to specify the order of the attributes for a class. The *obis:defaultOrder* annotations create a compensating mechanism. An automated generation of these of axioms is built in OWLGrEd editor extension OWLGrEd/OBIS [14,18] for data ontology handling. So, with appropriate ontology editor support an IS developer would not notice the existence of these annotations.
- *obis:isAbstract*. Specifies that a class is abstract and that a possibility to create new instances as elements of this class is not to be included in UI.
- *obis:isDerived*. Specifies that a data or object property is derived and that a possibility to create new values or links corresponding to this property is not to be included in UI.
- *obis:view*. Specifies the presence and order of attributes (both data attributes and classifier attributes) shown for class instances within a table or form view. May include both attributes ascribed for the class and its subclasses in the

[4] Can be downloaded from viziquer.lumii.lv

[5] The ontology prefix obis: stands for the namespace http://obis.lumii.lv/2013/01/obis#

ontology, as well as any other attributes (e.g. attributes whose domain (ascription) in the ontology is not defined). The *obis:view* annotations, where specified, take precedence over *obis:defaultOrder* annotations and it is in general considered that the *obis:view* annotations are to be entered manualy.

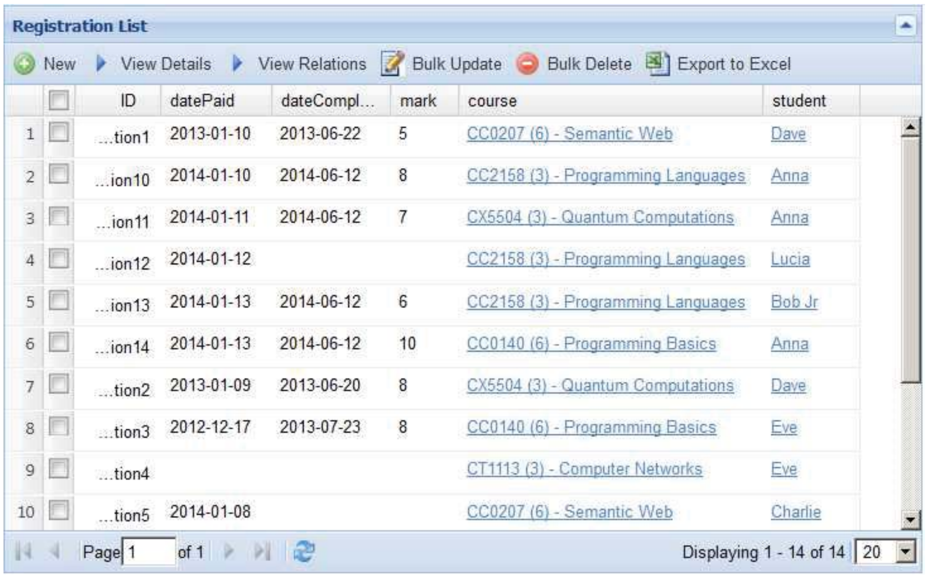
Figure 5 shows the mini-University ontology of Figure 1 extended with some example user interface annotations added. The notation used is domain specific ontology notation in the OWLGrEd ontology editor [18], it presents the *obis:textPattern* annotation in a textual form, *obis:isEnumerated* as <<EnumClass>> stereotype and *obis:isAbstract* as italics font face of the class name. The *obis:defaultOrder* annotations are created automatically by the OWLGrEd/OBIS editor and are not seen by the ontology end user.



```
AnnotationAssertion(obis:defaultOrder :Nationality "nCode,nValue")
AnnotationAssertion(obis:isEnumerated :Nationality "true")
AnnotationAssertion(obis:textPattern :Nationality "{nCode} - {nValue}")
AnnotationAssertion(obis:isAbstract :Person "true")
AnnotationAssertion(obis:defaultOrder :Person "personName,personID,nationality")
```

Fig. 5. Mini-University ontology with obis: annotations and annotation examples in OWL Functional Syntax

Figure 6 shows the effect of the introduced text pattern annotations on the presentation of Registration class in the generated OBIS application. Note that without the "linking in" the course and student information, the presentation of the Registration table would appear rather non-informative. The links behind the instances in the table open the detail views of the corresponding referenced instances. Notice that the generation of the illustrated user interface has been possible with just a minor manual tuning of the ontology (introducing *obis:textPattern* annotations to the *Student* and *Course* classes).


Registration List

New | View Details | View Relations | Bulk Update | Bulk Delete | Export to Excel

		ID	datePaid	dateCompl...	mark	course	student
1	☐	...tion1	2013-01-10	2013-06-22	5	CC0207 (6) - Semantic Web	Dave
2	☐	...ion10	2014-01-10	2014-06-12	8	CC2158 (3) - Programming Languages	Anna
3	☐	...ion11	2014-01-11	2014-06-12	7	CX5504 (3) - Quantum Computations	Anna
4	☐	...ion12	2014-01-12			CC2158 (3) - Programming Languages	Lucia
5	☐	...ion13	2014-01-13	2014-06-12	6	CC2158 (3) - Programming Languages	Bob Jr
6	☐	...ion14	2014-01-13	2014-06-12	10	CC0140 (6) - Programming Basics	Anna
7	☐	...tion2	2013-01-09	2013-06-20	8	CX5504 (3) - Quantum Computations	Dave
8	☐	...tion3	2012-12-17	2013-07-23	8	CC0140 (6) - Programming Basics	Eve
9	☐	...tion4				CT1113 (3) - Computer Networks	Eve
10	☐	...tion5	2014-01-08			CC0207 (6) - Semantic Web	Charlie

Page 1 of 1 | Displaying 1 - 14 of 14 | 20

Fig. 6. A class view including the linked instance descriptions.

The option of fine-tuning the OBIS application after its initial generation is based on the principle of availability of the OBIS application configuration also as OBIS application itself, so OBIS itself can be used to perform its application fine-tuning.

4 Discussion and Conclusions

The OBIS Framework demonstrates the feasibility of automated generation of feature-rich information systems just from annotated data ontologies. Besides a potential stand-alone usability for viewing or managing data structured in accordance to the RDF format, the concepts from OBIS might appear useful in the context of other editors of instances corresponding to structure defined by OWL ontology or RDF Schema, such as Web Protégé [19] or TopBraid Ensemble [20] that aim at similar ontology instance editing form functionality.

Although the OBIS tool can be used to interpret data ontologies created by different tools, its integration within the OWLGrEd tool [16] and its OBIS-extension [18], [14] may appear beneficial for the ease of information system customization in OBIS. We note also an important use case of OBIS in viewing RDF data corresponding to conceptual models of data coming from relational databases [14].

There is work in progress towards creating the custom-built reports, extending the validation rule set, as well as extending the set of annotations, interpretable by OBIS, however, trying to keep the annotations to be specified by the user to a comprehensible minimum. The advances towards creating user-comprehensible UI annotations that allow generation of fully functional UI from the annotated data model could be of value also for UI generation from UML-style conceptual models.

Another possible venue of the OBIS extension is via integration of the visual SPARQL query creation tool ViziQuer [17], however, this option requires a technological re-work of the ViziQuer tool via placing it into web-based framework.

References

1. Resource Description Framework (RDF), http://www.w3.org/RDF/
2. RDF Schema [WWW] http://www.w3.org/TR/rdf-schema/
3. Motik, B; Patel-Schneider P.F; Parsia B.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax, 2009
4. SPARQL 1.1 Overview. W3C Recommendation 21 March 2013 [WWW] http://www.w3.org/TR/sparql11-overview/
5. Linked Data, http://linkeddata.org
6. Tim Berners-Lee, James Hendler and Ora Lassila, "The Semantic Web", Scientific American, May 2001, p. 29-37.
7. R2RML: RDB to RDF Mapping Language [WWW] http://www.w3.org/TR/r2rml/
8. C.Blakeley: "RDF Views of SQL Data (Declarative SQL Schema to RDF Mapping)", OpenLink Software, 2007.
9. D2RQ Platform. Treating Non-RDF Relational Databases as Virtual RDF Graphs. http://www4.wiwiss.fu-berlin.de/bizer/D2RQ/spec/
10. Bagosi, T., Calvanese, D., Hardi, J., Komla-Ebri, S., Lanti, D., Rezk, M., Rodriguez-Muro, M., Slusnys, M., & Xiao, G. (2014). The Ontop framework for ontology based data access. In Zhao, D., Du, J., Wang, H., Wang, P., Ji, D., & Pan, J. Z. (Eds.), CSWS 2014, Vol. 480 of Communications in Computer and Information Science, pp. 67-77. Springer.
11. K.Čerāns, G.Būmans, RDB2OWL: a RDB-to-RDF/OWL Mapping Specification Language // J.Barzdins and M.Kirikova (eds.), Databases and Information Systems VI, IOS Press 2011, p.139-152.
12. Stardog, http://stardog.com/
13. G.Barzdins, E.Liepins, M.Veilande, M.Zviedris: Semantic Latvia Approach in the Medical Domain. // Proc. 8th International Baltic Conference on Databases and Information Systems. H.M.Haav, A.Kalja (eds.), TUT Press, pp. 89-102. (2008).
14. K. Cerans, G. Barzdins, G. Bumans, J. Ovcinnikova, S. Rikacovs, A. Romane and M. Zviedris. A Relational Database Semantic Re-Engineering Technology and Tools // Baltic Journal of Modern Computing (BJMC), Vol. 3 (2014), No. 3, pp. 183-198.
15. M.Zviedris, A.Romane, G.Barzdins, K.Cerans. Ontology-Based Information System // Proceedings of JIST'2013, LNCS, Vol. 8388, 2014, pp. 33-47.
16. Barzdins, J.; Barzdins, G.; Cerans, K.; Liepins, R.; Sprogis, A.: OWLGrEd: a UML Style Graphical Notation and Editor for OWL 2. In Proc. of OWLED 2010, 2010.
17. Zviedris M., Barzdins G. (2011), ViziQuer: A Tool to Explore and Query SPARQL Endpoints // The Semantic Web: Research and Applications, LNCS, 2011, Volume 6644/2011, pp. 441-445
18. K.Čerāns, J.Ovčiņņikova, R.Liepiņš, A.Sproģis, Advanced OWL 2.0 Ontology Visualization in OWLGrEd // A.Caplinskas, G.Dzemyda, A.Lupeikiene, O.Vasilecas (eds.), Databases and Information Systems VII, IOS Press, Frontiers in Artificial Intelligence and Applications, Vol 249, 2013, pp.41-54
19. WebProtege, http://webprotege.stanford.edu
20. TopBraid Ensemble, http://www.topquadrant.com/product/TB_Ensemble.html

Context-Dependent App Testing

Tim A. Majchrzak[1] and Matthias Schulte[2]

[1] University of Agder, Kristiansand, Norway
[2] viadee Unternehmensberatung GmbH, Münster, Germany
tima@ercis.de, Matthias.Schulte@viadee.de

Abstract. Software quality of apps often is low, which at least partly results from problems with testing them. A main problem are frequent context changes that have to be dealt with. Network parameters such as latency and usable bandwidth change while moving; usage patterns vary. To address context changes in testing, we propose a novel concept. It is based on identifying blocks of code between which context changes are possible. It helps to greatly reduce complexity.

Keywords: app, mobile, test, testing, context

1 Introduction

Mobile applications – *apps* – draw greatly from the possibilities offered by devices such as smartphones and tablets, e.g. by making use of localization via GPS. *Software testing* is a main challenge in app development. Testing software is a cumbersome task [13] and requires sophistication. App testing poses several particularities: Apps are not developed on the platform they run (mobile device) but on a PC. Testing on emulators will not yield the same results as testing natively. This also makes it laborious and hard to automate. Moreover, tool support currently is limited. Finally, many apps combine various technologies and programming languages.

Testing is greatly influenced by *context*. Mobile devices are subject to many different contexts; the simplest one is location, since mobility typically means frequent changes of position. There are many further context changes such as network condition, availability of data from sensors, and even social issues such as sharing devices. Thus, devices need to be tested taking context into considerations.

We propose a novel approach for software testing of apps that takes into account context changes. Our contributions are a sketch of context as an influencing factor of apps, the introduction of our unique approach for handling context in testing, and the demonstration of a real-life scenario to underline feasibility.

This paper is structured as follows. Section 2 highlights the relevance of context changes. Section 3 explains our approach. Application, limitations and challenges are discussed in Section 4.

2 Mobile Devices and Context

The usage paradigm of apps contrasts that of applications on PCs and that of Web sites (Webapps). The main difference is *mobility*. Even if an app not explicitly

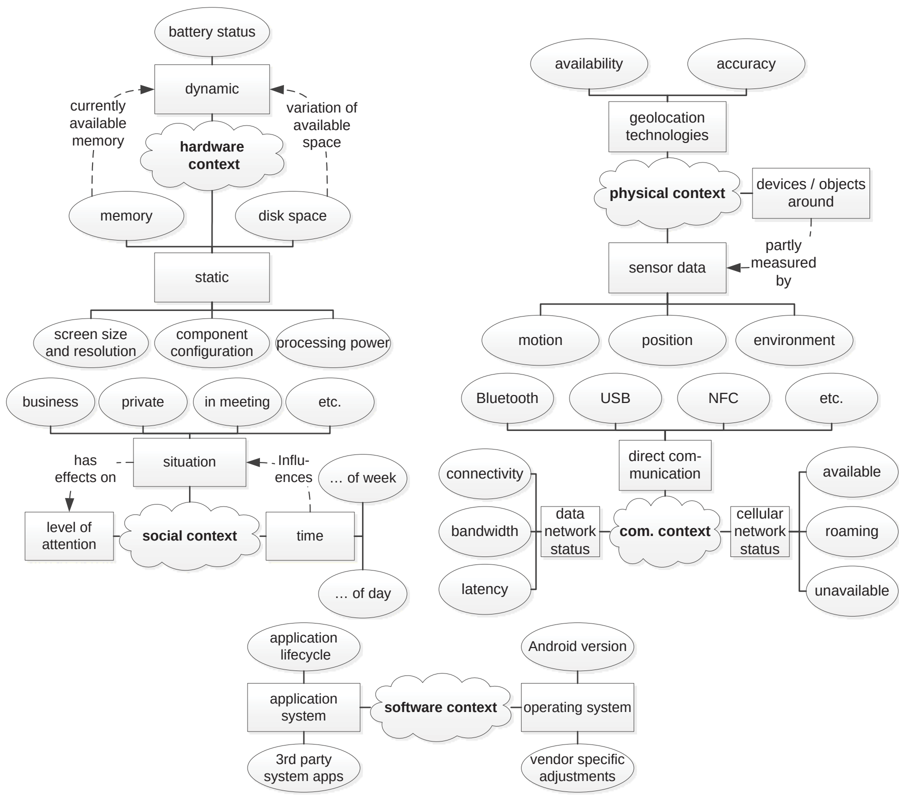


Fig. 1. Contexts

takes notice of mobility it is influenced by it due to changing conditions e.g. regarding connectivity. Apps might be halted on external events such as incoming phone calls. Moreover, they are built to be used in a changing environment.

Consequently, apps are heavily influenced by *context* but are also capable of making use of it. To give an example: when you drive out of town your smartphone might need to switch to a cellular system providing less bandwidth (adaption to context) but help you find a nearby swimming lake by matching geolocation information with map data (utilization of context).

2.1 Contexts Relevant for Mobile Devices

In the following, we propose a categorization scheme for app context that distinguishes five core contexts. It is sketched in Figure 1.

Firstly, there is the *hardware context*. Apps run on a multitude of devices, ranging from *smartwatches* to TVs. Components of devices greatly differ, leading to various screen sizes, resolutions, sensors, and input means besides touch to name just a few. Apps might not find the kind of hardware they require for proper operation and might need to adapt to hardware of different quality. Additionally, available memory and battery capacity have to be taken into consideration. The situation is worsened by the rapid progress in hardware development, making requirements very hard to forecast. The problem can be handled by seeking for a kind of "greatest common divisor" among devices. Context testing should include various possibilities of encountered hardware, e.g. with a sensor being present or not, and considering different levels of quality (such as precision). A testing strategy similar to equivalence partitioning [13, p. 28] might be chosen.

Fragmentation in terms of hardware goes along with software fragmentation, forming the *software context*. With at least four popular platforms [9], each existing in a variety of versions and some even modified by the hardware vendors, app development is problematic. It is unlikely that soon a cross-platform approach [12] will alleviate the problem. In fact, apps existing both in a native version and as a mobile Webapp even complicate testing. Context-related problems might also arise from the combination of contextual factors of hardware and software.

Mobility means that the *physical context* is continuously changing [18]. Location determines factors such as connectivity [19]. Most devices have one or more means to determine their location. The physical context also comprises other devices that can be contacted with technology such as Bluetooth and Near Field Communication (NFC). Moreover, devices are usually equipped with a number of context-depended sensors such as gyroscopes, thermometers and similar units.

Depending on the location, connection parameters such as availability, bandwidth and latency vary. This forms the *communication context*. it is influenced by static (e.g. the carrier of a user's choice) and dynamic factors. Location typically determines which mobile services are available and which bandwidth can be used. Apps ought to be robust and maintain functionality in offline scenarios. Testing includes simulations of small bandwidths, high latencies, changes in connection quality, and abrupt unavailability of service as well as resuming service.

The *social context* is harder to grasp than the other context categories. It comprises of user-specific ways of using an app. Firstly, more and more mobile devices are used both for work and for private purposes as part of *Bring your own device* (BYOD) [7] policies. Some apps are used for work, others are used for personal reasons, and some for both (but probably used in a different way). Secondly, some devices are used by more than one person. Different people use apps uniquely despite there typical set-up single-user scenarios. Thirdly, users' attention span will not be the same in all situations. This can be explained with the mobile nature: a person that uses the smartphone while walking would risk bumping into a street light if constantly staring on the screen. The social context is very challenging for testing; its factors are fuzzy, hard to estimate and in many cases impossible to exhaustively simulate.

In consequence, app testing has to be adjusted. Frequent changes of context have to be expected and patterns of change not necessarily are predictable. The multiplicativity of contexts makes testing more complex and time-consuming.

2.2 Related Work

The relevance of context in mobile computing has been discussed as early as in 2001 [16]. Despite varying notations, context is often used in connection with awareness, i.e. devices' ability to perceive changes and react accordingly [4, 20].

Standard testing literature is valuable since Web-based applications are typically covered (e.g. [17, Chap. 22]). Techniques for testing of graphical user interfaces (GUI) can be applied to apps. Mobility is sometimes covered. Factors such as different connection speeds of mobile services [15, pp. 166] might be addressed despite not explicitly discussing context. However, textbooks on app development often do not address testing but for some exceptions such as [14].

There is a variety of – typically research-in-progress – papers on app testing. Directions of research are automation [8], user-centered testing [11], tools [10], approaches [2], and user interface testing [21, 5]. All these papers tackle testing of apps but are conceptually different to our approach. In a work complementary to ours, Amalfitano et al. [1] consider context as the result from events triggered by the user, the phone, or external activities. They propose to identify patters and use them for manual, mutation-based and exploration-based testing.

3 Context-Sensitive Testing

To combine context changes with app testing, it has to be possible to automatically change context parameters whilst test cases are executed. The naive approach would be to specify the context for each test case a priori. Asserting that a single code unit produces an expected output in a certain context would not be helpful for testing business processes, though. As many influencing contextual factors are not static but change constantly during usage, a dynamic approach is needed.

Being able to specify context changes still is static if they have to be stated *within* test cases. For scenarios including different variations of context many test cases are required, each containing the same test code. This multiplication is not desirable. Test cases would be costly to develop and hard to maintain.

3.1 General Considerations and Principles

To provide a solution that fosters dynamic changes of context, we use modularization. Test cases are split into *blocks*; between each block a *change of context* is possible. Blocks are reused and combined with context changes. Therefore, testing different scenarios is possible without duplication of test code. As blocks are the foundation of our concept, we call it *block-based context-sensitive testing.*

A given test case may result in a number of blocks, each containing *operations* and *assertions*. Similar to unit testing frameworks, operations are needed to simulate user interaction; assertions are used to verify expected behavior. Our idea is to derive blocks from existing test cases. A single case may be transformed into a structure of blocks that can be used to generate context-dependent ones. To preserve the test case's intention, blocks are ordered and executed consecutively.

Listing 1.1 illustrates in a schematic way how a test case looks like. If the test code itself would be divided into blocks, this schematic test case contains two of them: one from lines 3 to 7 (*block A*) and one from lines 10 to 12 (*block B*).

The scope of each block has to be atomic w.r.t. changing contexts. In other words, during execution of test operations belonging to one single block, the

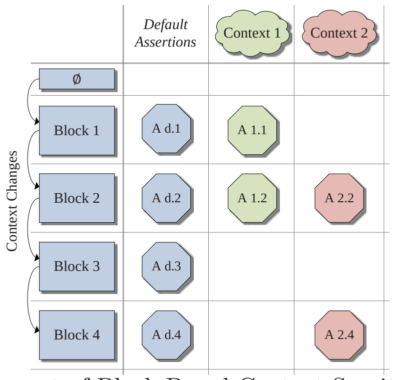


Fig. 2. Concept of Block-Based Context-Sensitive Testing

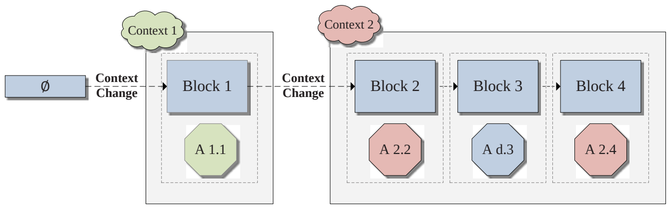


Fig. 3. Example of Test Execution Using Block-Based Context-Sensitive Testing

context remains stable. Only between blocks changes of context are possible (lines 2 and 9). To realize different scenarios, only the context changes in between have to be altered. The solution is expected to be most beneficial utilizing a generator that automatically creates test cases from blocks. Manual effort for writing context-sensitive test cases is reduced and redundant test code minimized.

```
 1 public void testExample(){
 2   contextChange(contextA);
 3   clickSomewhere();
 4   enterText();
 5   clickButton();
 6   ...
 7   assertThat();
 8
 9   contextChange(contextB);
10   doSomeOtherThings();
11   ...
12   assertThat();}
```

Listing 1.1. Schematic Test Case with Context Changes

Assertions have to be extracted from blocks due to their potential dependency on a context. For each block at least one *default assertion* has to be assigned, which verifies the app's standard behavior. For each known context, a specific assertion may be defined to assess context-dependent behavior. The blocks shown in the schematic test case in Listing 1.1 therefore have to be tailored smaller. Strictly spoken, assertions in lines 7 (Block A) and 12 (Block B) are not part of the blocks but have to be treated as another building block of our concept. Depending on the app's expected behavior, assertions may be *default* or *context-sensitive*.

The building blocks of block-based context-sensitive testing are shown in Figure 2. The structure of a test case is depicted by an ordered list of blocks.

3.2 Context-Dependent Test Case Execution and Example

A possible test execution is illustrated in Figure 3. The beginning of the test case is denoted as the empty set. Before the first block is executed, `Context 1` is established. Next, an assertion fitting to the current context is searched: there is one for `Context 1`. As the assertion holds, execution continues. Between `Block 1` and `Block 2` the context is changed again. This time, `Context 2` is established and kept for the remaining execution. The second block is executed similarly to the first one. This changes when reaching `Block 3`, which does not have any specific assertions. Consequently, the expected behavior is invariant between various contexts and the default assertion is evaluated. Finally, `Block 4` is executed together with assertion `A 2.4`, which is the assigned assertion for `Context 2`.

As shown in the matrix in Figure 2, there are numerous execution paths as prior to each block the context is changeable and alternative assertions can be defined. The matrix grows with the number of contexts but typically is sparse.

To illustrate practical benefits, we implemented a proof-of-concept tool and evaluated the approach in cooperation with an industry partner. We used a simple app to explain how it works: a client for the micro blogging service *Twitter*, which allows logging in to the service and posting messages. To prepare the setting, a jar archive containing our proof-of-concept has to be added to the *classpath* of the app's Android testing project. Moreover, as the Internet connectivity has to be changed to test the app in various contexts, `ACCESS_NETWORK_STATE` and `CHANGE_NETWORK_STATE` permissions have to be added to the `AndroidManifest.xml` of the app if not already contained.

The process steps for testing are logging in with invalid credentials, logging in with correct credentials, and posting a message. Each step can be conducted in a different context. Listing 1.2 shows the first block implemented with our solution. The test operations are implemented in the operation part of the block (lines 4 to 7). The credentials are invalid: the app is expected to show a corresponding message. This is checked by the default assertion (lines 8 to 10). However, if the app is in *disconnected* context, it is expected to show an error message. This context-dependent behavior is verified by an alternative assertion (lines 14 to 16).

```
 1 Context discContext = new Context(ConnectionStatus.DISCONNECTED);
 2
 3   Block loginIncorrectBlock = new Block(){
 4     public void operation() {
 5         enterText(usrName, "dummy@user.de"); enterText(pwd, "1234");
 6         clickOnButton(0);
 7     }
 8     public void defaultAssertion() {
 9       assertTrue(waitForText("Authentication␣failed!"));
10     }
11   };
12
13   loginIncorrectBlock.addAlternativeAssertion(discContext, new ↩
         Assertion() {
14     public void assertion() {
15       assertTrue(waitForText("Could␣not␣login:␣No␣connection."));
16     }
17   });
18 }
```

Listing 1.2. Sample Block implementation

For each of the above stated process steps a block with context-specific assertions is implemented and added to a list of blocks. A generator creates test cases as different mutations in terms of context changes from that list. In one possible test case, the first two blocks are executed in connected context and their default assertions are used for verifying. Before executing the third block, the context is changed. The context dependent assertion for that block checks whether in the *disconnected* context the message "No connection" is shown. Even in a small scenario as shown here a lot of different execution paths are possible: testing *without* our approach would be burdensome.

We used two sample generators to create test cases from the blocks explained above. In each test case the generator changes the context at different steps in the process. Finally, test cases are executed by means of JUnit. Like any other test for the Android platform, they are running on the device or emulator itself.

4 Discussion and Conclusion

In this paper we presented block-based context-sensitive testing. Our concept extends the literature of context-sensitivity in mobile computing. We have shown the viability of our approach in a case study and by presenting a prototype.

Out proof-of-concept Android tool is written in Java and can be checked out from GitHub [6]. Release under the Apache License allows free usage and modification; studying the source code will allow rapid development of similar tools for platforms such as Apple iOS. While the tool is not a core contribution of our work, it demonstrates the feasibility of our approach.

We found that by using our proof-of-concept it is possible to test apps in various contexts effectively. Blocks of test code are reusable and thus the effort in writing tests is reduced. Using generators the concept even gets more effective. Testers only have to implement blocks and define context-specific assertions once. Our approach also helps to find errors in code units where they are not expected.

However, our concept cannot (yet) be a panacea. We deem it a supportive mean for testing parts of apps which are heavily influenced by context. Due to the novelty of our approach, several limitations exist. Firstly, having a ordered linear list of blocks allows for only one way of execution, but the order of blocks and the decision if a block is executed may depend on the context used. We introduced a method for aborting test case execution to cope with this. Anyhow, our concept is not able to deal with process steps that *only* have to be executed in certain contexts. Secondly, the case study is an example of feasibility, yet not exhausting. Effectiveness has to be proven both qualitatively and quantitatively. Thirdly, our prototype is scarcely commented and not yet very user friendly.

These limitations have to be kept in mind yet do not question the general feasibility of our approach. In fact, they lead to future work. The most important task is an extension of our tool. Better support for test case generation w.r.t. context selection is desirable. Compiling best practices for context-sensitive testing would complement our work. A general challenge is to become able to cope with ample contexts. While our approach theoretically is capable of dealing with arbitrary context changes, actually simulating them for testing is very hard.

Future work needs to refine our concept and investigate into the consequences of context changes. Moreover, we will scrutinize the relationship of our work to other methods, e.g. *data-driven testing* [3]. We will also have a look at different domains that might share properties important for testing. For example, the context-dependence described here might similarly be found in embedded systems.

Acknowledgments

We would like to thank viadee Unternehmensberatung GmbH for their support.

References

1. Amalfitano, D., Fasolino, A.R., Tramontana, P., Amatucci, N.: Considering context events in event-based testing of mobile applications. In: Proc. ICSTW. pp. 126–133. IEEE CS (2013)
2. Anand, S., Naik, M., Harrold, M.J., Yang, H.: Automated concolic testing of smartphone apps. In: Proc. ACM SIGSOFT FSE '12. pp. 59:1–59:11. ACM (2012)
3. Baker, P., Dai, Z., Grabowski, J., Haugen, Ø., Schieferdecker, I., Williams, C.: Data-driven testing. In: Model-Driven Testing, pp. 87–95. Springer (2008)
4. Böhmer, M., Lander, C., Krüger, A.: What's in the apps for context? In: Proc. 2013 UbiComp Adjunct Pub. pp. 1423–1426. ACM (2013)
5. Choi, W.: Automated testing of graphical user interfaces: A new algorithm and challenges. In: Proc. ACM WS on MobileDeLi. pp. 27–28. ACM (2013)
6. contextTesting @GitHub (2014), https://github.com/viadee/contextTesting
7. Disterer, G., Kleiner, C.: Using mobile devices with BYOD. Int. J. Web Portals 5(4), 33–45 (2013)
8. Gao, J., Bai, X., Tsai, W.T., Uehara, T.: Mobile application testing. Computer 47(2), 46–55 (2014)
9. Gartner Press Release (2012), http://www.gartner.com/it/page.jsp?id=1924314
10. Gomez, L., Neamtiu, I., Azim, T., Millstein, T.: Reran: Timing- and touch-sensitive record and replay for android. In: Proc. ICSE '13. pp. 72–81. IEEE Press (2013)
11. Haller, K.: Mobile testing. SIGSOFT Softw. Eng. Notes 38(6), 1–8 (Nov 2013)
12. Heitkötter, H., Hanschke, S., Majchrzak, T.A.: Evaluating cross-platform development approaches for mobile applications. In: LNBIP, vol. 140, pp. 120–138. Springer (2013)
13. Majchrzak, T.A.: Improving Software Testing. Springer, Heidelberg (2012)
14. Milano, D.T.: Android application testing guide. Packt (2011)
15. Nguyen, H.Q.: Testing Applications on the Web. Wiley (2003)
16. Nugroho, L.E.: A context-based approach for mobile application development. Ph.D. thesis, Monash University (2001)
17. Perry, W.: Effective methods for software testing. Wiley, New York, 3rd edn. (2006)
18. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: Proc. of the 1994 1st WMCSA. pp. 85–90. IEEE CS (1994)
19. Schmidt, A., Beigl, M., Gellersen, H.W.: There is more to context than location. Computers & Graphics 23(6), 893–901 (1999)
20. Taranu, S., Tiemann, J.: General method for testing context aware applications. In: Proc. 6th Int. Workshop on MUCS. pp. 3–8. ACM (2009)
21. Yeh, C.C., Huang, S.K., Chang, S.Y.: A black-box based android GUI testing system. In: Proc. 11th MobiSys. pp. 529–530. ACM (2013)

RDB2OWL: A Language and Tool for Database to Ontology Mapping

Kārlis Čerāns[1], Guntars Būmans[1]

karlis.cerans@lumii.lv, guntars.bumans@gmail.com
Institute of Mathematics and Computer Science, University of Latvia
Raina blvd. 29, Riga, LV-1459, Latvia

Abstract. We demonstrate a high-level RDB2OWL language for database-to-ontology mapping specification in a compact notation within the ontology class and property annotations, as well as a corresponding mapping implementation tool that translates the RDB2OWL mappings into executable D2RQ mappings thus allowing to create a semantic SPARQL endpoint containing conceptually organized data from a relational database.

Keywords: Relational databases, OWL ontologies, RDF schemas, Database to ontology mappings.

1 Introduction

Exposing the contents of relational databases (RDB) to semantic web standard RDF [1] and OWL [2] formats enables the integration of the RDB contents into the Linked Data [3] and Semantic web [4] information landscape. An important benefit of RDB-to-RDF/OWL mapping is also the possibility of creating a conceptual model of the relational database data on the level of RDF Schema/OWL and further on accessing the RDB contents from the created semantic/conceptual model perspective. Since the RDB-to-RDF/OWL mapping connects the technical data structure (the existing RDB schema) with the conceptual one, it would typically have not a one-to-one data structure reproduction, but would rather involve some data transformation means.

The technical task of mapping relational databases to RDF/OWL formats is nowadays well understood and widely studied, just to mention D2RQ [5], Virtuoso RDF Graphs [6], Ultrawrap [7], Spyder [8] and ontop [9] among different RDB-to-RDF/OWL mapping languages and tools, together with W3C standard R2RML [10]. The languages of most RDB-to-RDF/OWL mapping approaches offer conceptually clear mapping structure with less attention paid, however, to the concise mapping writing possibility suitable both for manual mapping information creation as well as for using the mapping information as semantic-level documentation of the RDB structure. We review here a slightly simplified variant of the RDB2OWL mapping

[1] Partially supported by Latvian State Research programme NexIT project No.1 'Technologies of ontologies, semantic web and security'

language [11,12,13] that allows creating RDB-to-RDF/OWL mapping specifications in a compact textual form with mapping descriptions placed in the annotations of the target ontology entities. The RDB2OWL language allows re-using both the target ontology and source database structure information within the mapping definition, as well as introducing user defined functions.

The central novel point of this demonstration is a tool implementing the RDB2OWL notation[2] via converting a RDB2OWL mapping into an executable D2RQ mapping that can further on be used by the D2RQ platform either to produce the RDF dump of the source RDB, or to turn it into a SPARQL endpoint. Alternative RDB2OWL implementations sharing the inner RDB2OWL mapping representation with the translation into D2RQ are underway.

A particular usage scenario for RDB2OWL mappings is a semantic re-engineering of existing relational databases (cf. e.g. [14,15]) aiming at existing data access from the data ontology conceptual model perspective; this scenario has motivated most of the mapping constructions built into RDB2OWL. The tool has been used to re-work the Latvian medicine registries example [14] into a maintainable mapping solution (the original mapping solution of [14] involved creating custom SQL scripts just for the concrete database mapping; this approach allowed to receive the first principal proof-of-concept results, however, it did not appear practical when the database schema as well as mapping definition adjustments were to be involved).

Note that a textual RDB2OWL mapping specification can be used also as a reference for a (legacy) RDB by documenting the structure of the data contained therein.

2 RDB2OWL Mapping Language

A RDB2OWL mapping defines the correspondence between a relational database and an OWL ontology or RDF schema. The mapping information consists of elementary mappings or maps that are recorded in annotation assertions for ontology entities[3] - class maps for ontology classes, data property maps and object property maps for ontology data properties and object properties respectively.

The general pattern of class map as well as object and data property map definition follow the common sense structural principle of mapping ontology classes to database tables, data properties to table columns and object properties to links between tables.

More precisely, a class map in RDB2OWL is characterized by a table expression, possibly involving a join of several tables and a filter, and a pattern for instance resource URI construction from a table expression row. A class map can be attached to a single ontology/schema class, meaning that it describes instances for this class.

An object property map contains references to its subject (source) and object (target) class maps in the property map's (extended) table expression structure; such references bring the class map table expressions as components into the property map's table expression and provide the patterns of the property triple subject resp.

[2] The tool is freely available for download from `http://rdb2owl.lumii.lv/`

[3] We use `rdb2owl:DBExpr` as the annotation property, where `rdb2owl:=<http://rdb2owl.lumii.lv/2012/1.0/rdb2owl#>`

object URI generation from the property map's table expression row. An object property map is always attached to a single object property within the ontology.

Similarly, a data property map is characterized by a corresponding (extended) table expression that contains a reference to the property map's subject (source) class map, and a value expression describing the computation of the property literal value that corresponds to the URI, as computed by the subject class map's URI pattern. A data property map is always attached to a single data property within the ontology.

The textual form of RDB2OWL maps relies on textual presentations of table expressions, instance URI generation patterns and expression values; it is optimized to take into account the available ontology and RDB schema context information.

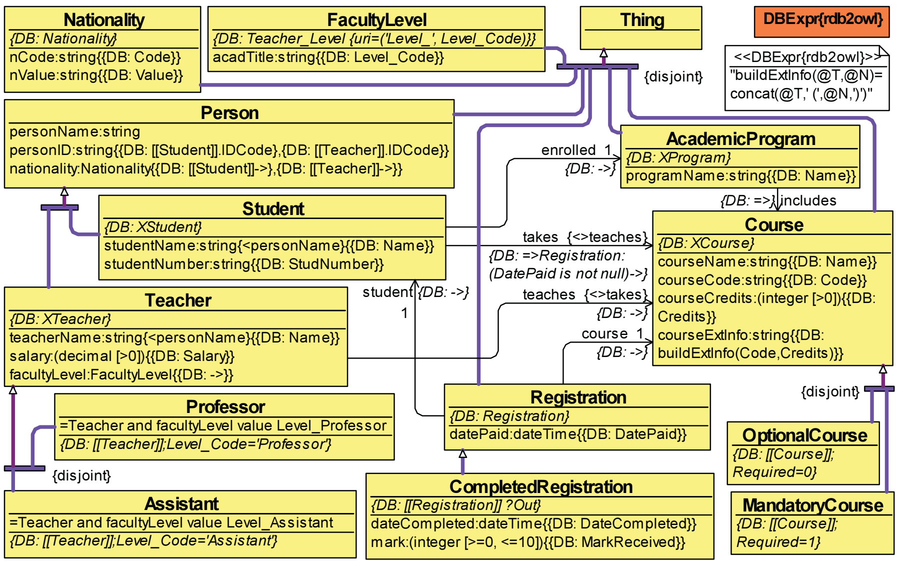
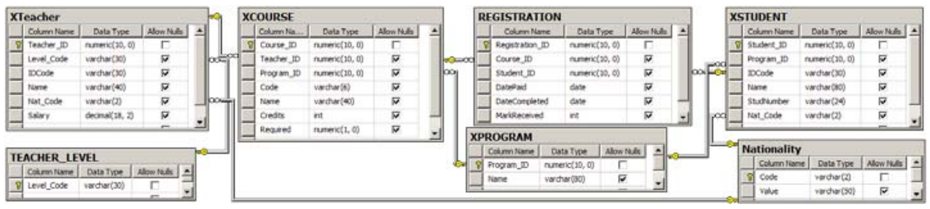

Nationality
{DB: Nationality}
nCode:string{{DB: Code}}
nValue:string{{DB: Value}}
FacultyLevel
{DB: Teacher_Level {uri=('Level_', Level_Code)}}
acadTitle:string{{DB: Level_Code}}
Thing
DBExpr{rdb2owl}
<<DBExpr{rdb2owl}>>
"buildExtInfo(@T,@N)=
concat(@T,' (',@N,')')"
{disjoint}
Person
personName:string
personID:string{{DB: [[Student]].IDCode},{DB: [[Teacher]].IDCode}}
nationality:Nationality{{DB: [[Student]]->},{DB: [[Teacher]]->}}
AcademicProgram
{DB: XProgram}
programName:string{{DB: Name}}
enrolled 1
{DB: ->}
{DB: =>} includes
Student
{DB: XStudent}
studentName:string{<personName}{{DB: Name}}
studentNumber:string{{DB: StudNumber}}
takes {<>teaches}
{DB: =>Registration:
(DatePaid is not null)->}
Course
{DB: XCourse}
courseName:string{{DB: Name}}
courseCode:string{{DB: Code}}
courseCredits:(integer [>0]){{DB: Credits}}
courseExtInfo:string{{DB: buildExtInfo(Code,Credits)}}
Teacher
{DB: XTeacher}
teacherName:string{<personName}{{DB: Name}}
salary:(decimal [>0]){{DB: Salary}}
facultyLevel:FacultyLevel{{DB: ->}}
student {DB: ->}
1
teaches {<>takes}
{DB: ->}
course 1
{DB: ->}
{disjoint}
Professor
=Teacher and facultyLevel value Level_Professor
{DB: [[Teacher]];Level_Code='Professor'}
Registration
{DB: Registration}
datePaid:dateTime{{DB: DatePaid}}
OptionalCourse
{DB: [[Course]];
Required=0}
{disjoint}
Assistant
=Teacher and facultyLevel value Level_Assistant
{DB: [[Teacher]];Level_Code='Assistant'}
CompletedRegistration
{DB: [[Registration]] ?Out}
dateCompleted:dateTime{{DB: DateCompleted}}
mark:(integer [>=0, <=10]){{DB: MarkReceived}}
MandatoryCourse
{DB: [[Course]];
Required=1}

Fig. 1. Mini-university ontology with full RDB2OWL mapping annotations
The '{DB: …}' notation is used for annotation assertions with rdb2owl:DBExpr property

In what follows, we outline the concrete syntax of RDB2OWL maps. Figure 1 contains an illustrative example of a mini-University OWL ontology in OWLGrEd[4] ontology editor notation [16,17] (e.g. showing OWL classes as UML classes, OWL object and data properties as roles on UML associations between property domain and range classes and attributes of property domain classes)[5]. The ontology in Figure 1 contains both its structure definition (the classes and properties) and further OWL constructs (e.g. class expressions and datatype facet restrictions); it is enriched by a custom *{DB: <annotation_value>}* notation for the database connection annotations (cf. [18]), as well. The corresponding RDB schema is outlined in Figure 2.

[4] `http://owlgred.lumii.lv/`

[5] The ontology with annotations in a standard OWL RDF/XML format is available at `http://rdb2owl.lumii.lv/demo/UnivExample.owl`



XTeacher

Column Name	Data Type	Allow Nulls
Teacher_ID	numeric(10, 0)	☐
Level_Code	varchar(30)	✓
IDCode	varchar(30)	✓
Name	varchar(40)	✓
Nat_Code	varchar(2)	✓
Salary	decimal(18, 2)	✓

XCOURSE

Column Na...	Data Type	Allow Nulls
Course_ID	numeric(10, 0)	☐
Teacher_ID	numeric(10, 0)	✓
Program_ID	numeric(10, 0)	✓
Code	varchar(6)	✓
Name	varchar(40)	✓
Credits	int	✓
Required	numeric(1, 0)	✓

REGISTRATION

Column Name	Data Type	Allow Nulls
Registration_ID	numeric(10, 0)	☐
Course_ID	numeric(10, 0)	✓
Student_ID	numeric(10, 0)	✓
DatePaid	date	✓
DateCompleted	date	✓
MarkReceived	int	✓

XSTUDENT

Column Name	Data Type	Allow Nulls
Student_ID	numeric(10, 0)	☐
Program_ID	numeric(10, 0)	✓
IDCode	varchar(30)	✓
Name	varchar(80)	✓
StudNumber	varchar(24)	✓
Nat_Code	varchar(2)	✓

TEACHER_LEVEL

Column Name	Data Type	Allow Nulls
Level_Code	varchar(30)	☐

XPROGRAM

Column Name	Data Type	Allow Nulls
Program_ID	numeric(10, 0)	☐
Name	varchar(80)	✓

Nationality

Column Name	Data Type	Allow Nulls
Code	varchar(2)	☐
Value	varchar(50)	✓

Fig. 2. A mini-University RDB schema

A table expression in RDB2OWL in the simplest and most common case is just a table name, such as *XTeacher*, or *Teacher_Level*; a filter expression can be added to a table expression, using a semicolon, such as '*XCourse;Required=1*'. The table expressions involving several tables can be formed either in

(a) item list notation with comma-separated, possibly alias-labeled table expressions, such as '*XTeacher T, XCourse C; T.Teacher_ID=C.Teacher_ID*', or
(b) navigation list notation, such as '*XTeacher[Teacher_ID]->[Teacher_ID] XCourse*' that can be shortened to '*XTeacher->XCourse*' by omitting navigation columns from table sole FKs to the matching PKs; the symbol => denotes PK-to-sole FK navigation, e.g. '*XProgram=>XCourse*', or
(c) notation combining both (a) and (b), where the whole navigation list can be regarded as a single item in the item list.

There is a further possibility of local filter introduction in navigation expressions, such as *XStudent=>Registration:(DatePaid is not null)->XCourse*.

The textual syntax for a class map consists of table expression and URI pattern specification, such as '*Teacher_Level {uri=('Level_', Level_Code)}*'; the URI pattern is expected to have a list of constant strings and column expressions whose values are concatenated to give the local name of the corresponding ontology class instance. The URI pattern can be omitted, if it is formed just from table expression's leftmost table name followed by its primary key column(s) value. A class map's table expression can refer also to a defined class map either by its explicit name, or by the name of the class for which it is the sole class map (e.g. *[[Teacher]]*).

The object property map is described as an extended table expression where two sub-expressions denote its subject (source) and object (target) class maps respectively. These sub-expressions can be either:

(a) explicitly marked by an alias <s> or <t>, respectively, in some place within the table expression's structure,
(b) defined by the mark <s> or <t> as a table expression item itself; in this case the sole class maps defined explicitly for the property domain or range class are considered the subject and object class maps for the property map, or
(c) in the case of missing explicit <s> and/or <t> marks these marks are assumed <s> for the leftmost and <t> for rightmost item within the table expression.

These conventions allow denoting e.g. the object property map simply by '->', if the property corresponds to the sole FK-to-PK mapping between the tables serving as table expressions in the sole class maps of property domain and range classes.

The data property map can be described by a column name/column expression that is to be evaluated in the context of the sole class map attached to the property domain class. Alternatively, <*table_expression*>.<*value_expression*> notation can be used for data property map description, where the table expression is required to contain either explicit or implicit reference <*s*> to the property map's subject class map.

The example in Figure 1 contains also an illustration of a function definition using an ontology level *rdb2owl:DBExpr* annotation, the defined function *buildExtInfo* is then used in a *courseExtInfo* property value expression for the *Course* class. The RDB2OWL functions can be used for repeating mapping fragments, as well as for increasing mapping readability in the case of complex mapping constructions. The function body can introduce also additional table expressions into the context of evaluating its value expression; such table expressions are joined to the table expression defining the context of evaluation of the function-calling value expression.

The decoration *?Out* in the database annotation for *CompletedRegistration* class means relating to this class only those rows of the described class map table expression that contain a value in at least one property whose domain is this class.

A more detailed RDB2OWL language design description can be found in [11] (the core concepts) and [12] (advanced constructs), as well as [13].

3 Mapping Implementation in RDB2OWL Tool

The RDB2OWL mapping tool[6] takes as the input annotated OWL ontology as well as a connection to the source database (in order to read the source database schema) and it produces a D2RQ mapping for accessing the source database via D2RQ platform[7] SPARQL endpoint or RDF dump tool. The database connection information (JDBC driver name and connection information) can be defined as an annotation in the data ontology, or it can be entered directly in the RDB2OWL tool.

The RDB2OWL mapping processing in the tool follows a number of steps activated either one-by-one or using a batch command "Create mapping from ontology":

- Load ontology: the ontology with the attached RDB2OWL annotations is loaded into the internal RDB2OWL mapping model [13]; the parsing of the annotations into the RDB2OWL model items is performed.
- Load source database schemas into the RDB2OWL model.
- Finalize abstract mapping: the advanced mapping constructs (e.g. named class maps, shorthand notation, defaults and user defined functions) are resolved into basic mapping constructions thus transforming the mapping into the semantic RDB2OWL model [13] outlined here in Figure 3.
- Generate D2RQ Mapping from the created semantic RDB2OWL model.

Although the navigation item and link construction can be translated into more basic reference item notation for table expressions corresponding to Raw RDB2OWL metamodel [11], both notations are retained in the RDB2OWL semantic model and the D2RQ mapping generation is performed directly from the notation involving the navigation items (corresponding to RDB2OWL Core MM of [11]). The model of Figure 3 is to be re-used also in upcoming alternative RDB2OWL implementations.

[6] http://rdb2owl.lumii.lv/; see this paper's example at http://rdb2owl.lumii.lv/demo/
[7] http://d2rq.org/

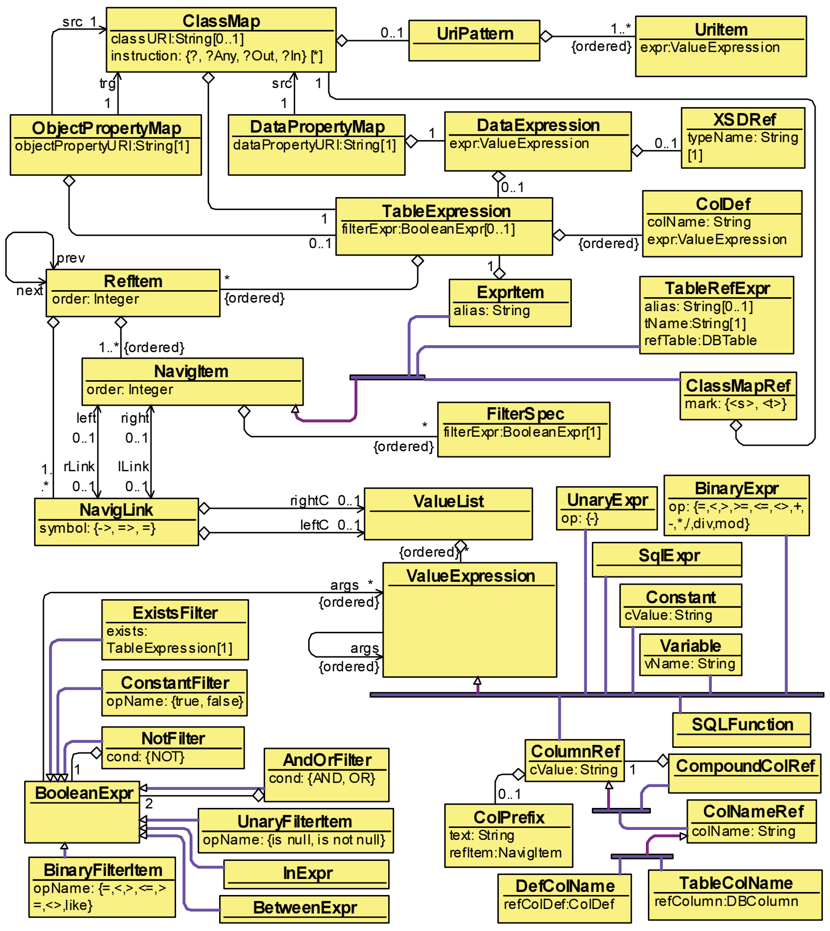


Fig. 3. Essential fragments of RDB2OWL semantic metamodel

An important parameter for the mapping generation is the “with inference” option that can be turned on or off by the user. We explain this parameter using an example. If an ontology class, say, *Student*, is said to correspond to a database table *XStudent*, one naturally expects that its superclass (e.g. *Person*) also automatically receives the instances coming from the *XStudent* table. If the RDF triples corresponding to the database are dumped and loaded into a RDF triple store (e.g. Virtuoso Server), the RDF triple store can perform the inference. If, however, the RDB data are to be accessed by D2R server on on-the-fly basis, this inference is built into the D2RQ mapping generation by the RDB2OWL tool so that together with the D2RQ class map for the *Student* class, a similar class map is generated also for the *Person* class (an alternative would be to extend the D2R server functionality with some inference capabilities). The RDB2OWL tool for D2RQ mapping generation currently supports subclass, subproperty and inverse properties inference. Similar inference capabilities are planned also for upcoming R2RML implementation.

The RDB2OWL tool is implemented in JAVA programming language, however, its current implementation uses libraries for the internal RDB2OWL model handling and transformations that are working only in MS Windows environment. The model transformations from RDB2OWL syntactic to semantic metamodel are written in LuA programming language using its lQuery model transformation library [19].

4 Discussion and Conclusions

The RDB2OWL mapping language and tool allows easy creation of database-to-ontology mappings involving basic one-to-one correspondence between ontology classes, data and object properties and database tables, columns and relations respectively, as well as involving more advanced constructs (e.g. adding filters, sub-class relations, value processing, etc.). The experience shows that for simple to medium size ontologies and corresponding databases the RDB2OWL language and tool, in combination with D2RQ Platform [5] and possibly an RDF triple store (in the case of initial database RDF dump created by the D2RQ server), allow to quickly obtain a SPARQL endpoint over conceptually structured view over RDB data up and running. The method is well applicable also if only a part on information from a larger database is to be exposed into the semantic form.

The practical experience of using RDB2OWL over larger ontologies and database schemas, as in Latvian medicine registries example of [14] with the ontology containing 173 classes, 219 object properties and 816 data properties, shows that the tool is well usable, however, some further ontology and mapping engineering means for cross-checking the validity and completeness of the created mappings would be helpful (these can be used both on the RDB2OWL annotation level, as well as on the level of the generated executable mapping code, e.g. in D2RQ).

The direct RDB2OWL mapping text size both in the considered mini-University and in the Latvian medicine registries [14] examples is about 8-9 times smaller, if compared with the generated D2RQ mapping text size, thus allowing considering RDB2OWL for manual database-to-ontology correspondence specification.

Regarding the OWL constructions allowed in the ontologies that are RDB2OWL mapping targets, we note that in fact, any OWL features also beyond the RDF Schema notation can be permitted. Figure 1 contains a number of such features, for instance equivalent class assertions and datatype facet restrictions. We can note that the equivalent class assertions for the subclasses of the *Teacher* class are “enforced” by the RDB2OWL mapping definition; while the facet restrictions in the current mapping version are left to be checked as constraints over the obtained RDF triple set.

Although the RDB2OWL tool has a well-defined task of creating a conceptual-level SPARQL endpoint for a relational database, permitting its standalone use, a particular usage context of the RDB2OWL database-to-ontology mapping tool is within a larger semantic database platform [15] involving also the OWLGrEd ontology editor [16,17], the visual custom SPARQL query generation tool ViziQuer [20] and the SPARQL endpoint browser OBIS [21].

References

1. Resource Description Framework (RDF), http://www.w3.org/RDF/
2. Motik, B; Patel-Schneider P.F; Parsia B.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax, 2009
3. Linked Data, http://linkeddata.org
4. Tim Berners-Lee, James Hendler and Ora Lassila, "The Semantic Web", Scientific American, May 2001, p. 29-37.
5. D2RQ Platform. Treating Non-RDF Relational Databases as Virtual RDF Graphs. http://www4.wiwiss.fu-berlin.de/bizer/D2RQ/spec/
6. C.Blakeley: "RDF Views of SQL Data (Declarative SQL Schema to RDF Mapping)", OpenLink Software, 2007.
7. Sequeda, J.F., Cunningham, C., Depena, R., Miranker, D.P. Ultrawrap: Using SQL Views for RDB2RDF. In Poster Proceedings of the 8th International Semantic Web Conference (ISWC2009), Chantilly, VA, USA. (2009)
8. Spyder tool, URL: http://www.revelytix.com/content/spyder
9. Bagosi, T., Calvanese, D., Hardi, J., Komla-Ebri, S., Lanti, D., Rezk, M., Rodriguez-Muro, M., Slusnys, M., & Xiao, G. (2014). The Ontop framework for ontology based data access. In Zhao, D., Du, J., Wang, H., Wang, P., Ji, D., & Pan, J. Z. (Eds.), CSWS 2014, Vol. 480 of Communications in Computer and Information Science, pp. 67-77. Springer.
10. R2RML: RDB to RDF Mapping Language, http://www.w3.org/TR/r2rml/
11. K.Čerāns, G.Būmans, RDB2OWL: a RDB-to-RDF/OWL Mapping Specification Language // J.Barzdins and M.Kirikova (eds.), Databases and Information Systems VI, IOS Press 2011, p.139-152.
12. G.Būmans, K.Čerāns, Advanced RDB-to-RDF/OWL mapping facilities in RDB2OWL // Proc. of BIR 2011, Riga, Latvia, October 7-8, 2011. LNBIP 90, pp. 142-157. Springer, Heidelberg, 2011 (ISBN:978-3-642-24510-7
13. G.Būmans, Relational database information availability to semantic Web technologies, Dr.sc.comp theses, University of Latvia, 2012, https://dspace.lu.lv/dspace/handle/7/2538.
14. G.Barzdins, E.Liepins, M.Veilande, M.Zviedris: Semantic Latvia Approach in the Medical Domain. // Proc. 8th International Baltic Conference on Databases and Information Systems. H.M.Haav, A.Kalja (eds.), TUT Press, pp. 89-102. (2008).
15. K. Cerans, G. Barzdins, G. Bumans, J. Ovcinnikova, S. Rikacovs, A. Romane and M. Zviedris. A Relational Database Semantic Re-Engineering Technology and Tools // Baltic Journal of Modern Computing (BJMC), Vol. 3 (2014), No. 3, pp. 183-198.
16. Barzdins, J.; Barzdins, G.; Cerans, K.; Liepins, R.; Sprogis, A.: OWLGrEd: a UML Style Graphical Notation and Editor for OWL 2. In Proc. of OWLED 2010, 2010.
17. Barzdins, J.; Cerans, K.; Liepins, R.; Sprogis, A.: UML Style Graphical Notation and Editor for OWL 2. In Proc. of BIR'2010, LNBIP, Springer 2010, vol. 64, p. 102-113.
18. K.Čerāns, J.Ovčiņņikova, R.Liepiņš, A.Sproģis, Advanced OWL 2.0 Ontology Visualization in OWLGrEd // A.Caplinskas, G.Dzemyda, A.Lupeikiene, O.Vasilecas (eds.), Databases and Information Systems VII, IOS Press, Frontiers in Artificial Intelligence and Applications, Vol 249, 2013, pp.41-54.
19. Liepiņš, R. Library for Model Querying – lQuery. In Proceedings of 2012 Workshop on OCL and Textual Modelling; 2012.
20. Zviedris M., Barzdins G., ViziQuer: A Tool to Explore and Query SPARQL Endpoints // The Semantic Web: Research and Applications, LNCS Volume 6644, 2011, pp. 441-445
21. M.Zviedris, A.Romane, G.Barzdins, K.Cerans. Ontology-Based Information System // Proceedings of JIST'2013, LNCS Volume 8388, 2014, pp. 33-47

The Influence of Syntactic Quality of Enterprise Process Models on Model Comprehension

Merethe Heggset[1], John Krogstie[1] and Harald Wesenberg[2]

[1]Norwegian University of Science & Technology - NTNU, Norway
merethhe@gmail.com, krogstie@idi.ntnu.no

[2] Statoil ASA, Norway
hwes@statoil.com

Abstract. In this paper we report on the use of process modelling in connection to the quality system of Statoil, a large Norwegian oil company, in particular on the aspects found necessary to emphasise to achieve the right quality of the models in this organisation. Based on investigation of usage statistics and user feedback on models, we have identified that many have trouble in comprehending some of the models. Many of these models have poorer syntactic quality than the average syntactic quality of models of the same size. An experiment with improving syntactic quality of one of these models is reported here. Further work is needed to look in more detail on the interplay between levels of quality for overall effect of enterprise models.

1 Introduction

Statoil is a Norwegian oil company with more than 23 000 employees and around the same number of external contractors. The company operates in 36 different countries and have in the last decade been using enterprise modelling in order to structure their vast amounts of organizational knowledge and information. They report to have achieved a fair amount of success with enterprise modelling in its corporate management system [13] where workflow models are used extensively to communicate requirements and best practices throughout the enterprise. The enterprise model functions as a common point of reference for the entire organisation, ensuring the quality of a large number of work processes and communicating requirements and best practices throughout the company. The models are used daily in large parts of the organization, and are a significant contributor in reducing operational, environmental and safety risks. As an example, the important SIF-index (Serious Injury Frequency) which counts the number of incidents per million work hours has been reduced from 6 to around 0.8 in the period since the models where introduced. Every week Statoil employees and contractors perform approximately 2 million work hours. That said, the process models are only one approach to risk mitigation. One also experience that the process models could be utilized even better.

A lot of research has been done in the field of enterprise process modelling, as well as on the subject of how to evaluate model quality [3, 6, 7, 8]. However, as stated by Moody [6], many of these methods suffer from a lack of adoption in practice. While the main goal of applying such frameworks in practice normally is providing a detailed evaluation of model quality in a specific case, it can also give indications of the usefulness of the framework.

From the start of the current modeling initiative, Statoil has been aware of the need to balance different levels of quality of the models. According to [10, 13] Statoil have found that it is useful to differentiate between at least three dimensions of model quality: Syntactic quality (how well the model uses the modelling language), semantic quality (how well the model reflects the real world) and pragmatic quality (how well the model is understood by the target audience), which are core dimensions of SEQUAL framework on quality of models and modelling languages [3]. SEQUAL builds on early work on quality of models, but has been extended based on theoretical results [6, 7, 8] and practical experiences [3, 4, 5] with various versions of the framework.

This paper present some of the results from a case study on the use of enterprise process models in Statoil, in particular looking upon model usage, quality issues of existing models, and in particular how better syntactic quality can influence the pragmatic quality of models. The main research question we have investigated in connection to this paper is” How do syntactic quality of a model influence on the pragmatic quality”.

In section 2 we describe the Statoil quality system in more detail, before we in section 3 describe experiences from evaluations of the current models, and an experiment on improving the syntactic quality of existing operational models. Discussion of results and ideas on further work are found in section 4.

2 Case Environment - Statoil Quality Management System

The enterprise model is realized through the Statoil management system. The Statoil Book [12], which is the foundation the management system is built upon, describes the management system as "the set of principles, policies, processes and requirements which support our organisation in fulfilling the tasks required to achieve our goals". It defines how work is done within the company, and all employees are required to act according to relevant governing documentation. The Management System consists of three main parts:

- Process models in ARIS, the modelling solution from which all governing documentation (GD) is accessed by the end users.
- Docmap, used for handling and publishing more detailed textual GD
- Disp, a tool which supports the process of handling applications for deviation permits in cases where compliance with a requirement is difficult or impossible to achieve.

The three main objectives of the Statoil Management System are

1. Contributing to safe, reliable and efficient operations and enabling compliance with external and internal requirements.
2. Helping the company incorporating their values, people and leadership principles into everything they do.
3. Supporting business performance through high-quality decision-making, fast and precise execution and continuous learning.

GD describes what is to be achieved, how to execute tasks, and ensures standardisation. Each process area has governing documentation in the form of documents and/or process models, accessible from the ARIS start page. A three-level process model structure is developed. On the bottom level, the so-called workflow diagrams, contains BPMN models [9] on the descriptive level. The quality system contains around 2000 BPMN models at this level, qualifying the case to be an example of BPMN in the large [2].

The enterprise process model is created according to a set of rules for structuring and use of notation that has evolved over the years of model development of use [10, 11]. Using the Splunk tool[1] one can capture how often a certain page or model is accessed and how users navigate through the enterprise model. 12 out of the 20 most used models represent safety critical processes, i.e. they are either classified as Safe work (a sub-category of Operation and Maintenance) or belong to the Safety process area.

When designing diagrams in the enterprise model, requirements in TR0002 - Enterprise structure and standard notation [11] shall be met. In [1], we provided a mapping of the Statoil modelling requirements on their version of BPMN [9] from TR0002 to SEQUAL. In the next section we will in particular look upon the current syntactic quality issues of models (including lacking conformance to naming and labelling guidelines which in [1] was listed under empirical quality).

3 Influence of Syntactic on Pragmatic Quality

During the end of 2013 and the beginning of 2014, a user survey was conducted in Statoil in order to better understand users' experiences and opinions related to the management system and governing documentation including the process models. The survey uncovered challenges regarding understanding of some of the models (pragmatic quality). Although a large proportion of user feel that the governing documentation is easy to understand, others report issues of vagueness and ambiguity.

One of the main purposes of the document TR0002 [11] is to ensure a high syntactic quality in the models made. The document provides an overview of the allowed symbols and naming conventions. The degree of syntactical correctness was first measured on seven workflow models. In the user survey, respondents were asked to give examples of processes that were interpreted differently within their department/unit. This list of processes was used as a basis when selecting models for evaluation. Due to a high number of models listed, not all could be evaluated. The following criteria were applied when selecting models:

[1] http://www.splunk.com/view/splunk/SP-CAAAG57

1. The process is directly mentioned by respondents in the user survey as a cause for misunderstandings and different interpretations
2. The total number of nodes and edges in the model is larger than 20.
3. The model is one of the 100 most used workflow models.

The rules are annotated according to the symbol or aspect they are related to, i.e.:

Table 1 : Syntactic quality measurements

Model	Size	Breaches	SYN	AVG
Apply for and evaluate work permit	21	7xN2, 2xG2, 2xG3, 2xN2, CA3	0,55	0,87
Prepare isolation plan	23	CA3, G2, N2	0,89	0,89
Project control	24	12xN4, N2,E1, CA2, CA4, G2, G3,SF1	0,48	0,82
Execute mechanical completion	30	2xN4, 4xN7, 4xE1, 3xW, 4x N2, 2xSF1, G2, G3	0,37	0,80
Set, verify and approve isolation	30	2xN2, 2x SF1, CA4	0,87	0,80
Safety incident	39	E1, 7xN2, 3xN2, 3xG2,2xG3, SP2, 4xW	0,58	0,78
Commissioning and handover of systems	46	2xE1, SP1, SP2, 2xSF1, 16x N4, 2xN2, 5xG2, 5xG3, 3xT1	0,39	0,78

- N: Naming conventions
- T: Task
- OT: Optional Task
- G: Gateways
- SP: Collapsed Sub Process
- CA: Collaboration Activity
- SF: Sequence Flow
- W: Wrongly used concept

The size of the model is equal to the total number of nodes (symbols) and edges (arrows). After measuring the syntactic quality (SYN) of the seven selected workflow models, they were compared to other models of a similar size. The criteria used when choosing models for comparison were the same as the criteria listed above, except for criteria 1 which was inverted - only models without direct mentions were included. For each of the "troublesome" models, the three models closest in size from the top 100 list, that also fit the set criteria were evaluated. The results are summarised in table 1, indicating errors the types found in the bullet list below.

- N1: Names on symbols and expressions shall be formulated in singular form
- N2: Avoid terms with more than four words if possible

- N3: A name shall not be a detailed description
- N4: The first letter of a symbol name shall be in upper case. All other letters should be lower case
- N5: Proper names shall start with upper case letters
- N6: The Statoil official name of a concept shall be used when alternatives exist
- N7: Abbreviations should be avoided
- T1: The title of a task shall be a verb imperative (reflecting the activity performed in order to add value), followed by a noun (reflecting the asset)
- OT1: The title of an optional task shall be a verb imperative (reflecting the activity performed in order to add value), followed by a noun (reflecting the asset)
- OT2: The use of an optional task is only allowed within a collaboration activity
- OT3: It is not allowed to connect sequence flows to the optional task symbol
- SP1: The title of a collapsed sub-process shall be a verb imperative (reflecting the activity performed in order to add value), followed by a noun (reflecting the asset)
- SP2: The collapsed sub-process symbol is drawn using a standard activity shape with a "+" attached
- CA1: The tasks grouped by a collaboration activity symbol shall not be sequenced in time or contain dependencies
- CA2: The title of a collaboration activity shall be a verb imperative (reflecting the activity performed in order to add value), followed by a noun (reflecting the asset)
- CA3: The name of a collaboration activity shall be unique and you shall not name the collaboration activity with names that have been used in the tasks that have been framed by the collaboration activity symbol
- CA4: Each of the tasks framed by the collaboration activity symbol must have a unique title, clarifying different type of activities performed by different roles
- E1: You shall define the title of a start or end event as a noun (reflecting the asset) followed by a verb past participle (reflecting the activity performed to add value to the asset)
- G1: You shall not name parallel gateways
- G2: The title of a diverging exclusive gateway shall consist of the term control (can be replaced with check, verify, evaluate or clarify) followed by a noun (reflecting the object submitted to control)
- G3: The exclusive flow shall be described through an adjective or a phrase describing the alternative flows. You shall not use yes or no when designing exclusive gateways
- SF1: A sequence flow shall have only one source and one target
- SF2: You should not use more than one sequence flow from an activity
- W: Using the wrong symbol (or similar errors)

4.1 Experiment Design and Results

In the experiment, two workflow models were selected, and changes were made to these models to increase their syntactic quality according to the guidelines described above. Participants were to answer questions related to the models in order to measure their understanding (pragmatic quality) of the models.

The original plan was to use only Statoil employees from different departments and locations as participants, but since it proved to be difficult to find enough volunteers, a student experiment was carried out in parallel. In total, 18 students and 9 Statoil employees participated in the study. In order to avoid participants answering based on personal knowledge rather than by consulting the models, the participants from Statoil were did not have first-hand experience with the modelled processes. The models selected for the experiment had a syntactic quality below average, and were found to be easily improvable by correcting mistakes according to the rules found in TR0002 [11]. The two models used in the experiment were:

- SF103 - Safety incident (see characteristics in table 1)
- OM05.07.01.03 - Reset isolation and pressurise

SF103 was also part of the syntactic quality evaluation reported above because it was highlighted in the user survey as a model subject to misinterpretations, and we focus on this below. (OM05.07.01.03 had syntactic quality on 0.72 i.e. around average).

Syntactic quality was here measured on the Norwegian versions of the models, as the experiment was conducted in Norwegian. This was decided in order to avoid language-related misunderstandings, as all of the respondents were native Norwegian speakers. When making the new versions, the models were adjusted to make the syntactic quality as close to 1 as possible. Major changes were made to SF103, as many of the errors were large, e.g. the wrong symbol was used in several cases. With OM05.07.01.03, the changes made were mostly corrections in naming of symbols and splitting of arrows.

The participants were each given two models to interpret - one original and one modified. The participants were split into four groups, and each group was given a different combination of models and questions, following a Latin square design. In addition, they were given an overview of the language notation. The participants were each given 15 questions connected to SF103, and 10 questions connected to OM05.07.01.03. When summarising the results, each wrongly answered question was given -1 points, unanswered questions were given 0 and correct answers were given a score of 1. The total number of available points for each model is the result of (number of participants x number of questions), e.g. 9 x 15 = 135 for questions to the old SF103 in the student experiment. Whereas few improvements were found on OM05.07.01.03, probably due to that the quality was sufficient; we look in more detail on SF103 below:

The overall results for SF103 are summarised in table 2. As shown, the modified version of SF103 scored much higher than the original version both in the Statoil experiment and the student experiment. Some specific questions are worth taking a closer look at, as they give insight into certain problem areas and normal misunderstandings. Question 2 stands out, as all of the Statoil participants answered it wrongly when looking at the old version of the model, and half of those looking at the new:

2. True or false: The process always starts with a safety incident occurring

Looking at the student respondents the change is even bigger: as many as 7 out of 8 that were given the original version answered the question wrongly, and only two that were given the new made the same mistake. The question is related to events, and in reality there are two possible triggers to the process. In the original version, many event-related symbols are used incorrectly, e.g. there are two cases of ”end event" symbols with sequence flows pointing out from them, and event symbols are used instead of task symbols even though the process does not start or end at these points.

Table 2 SF103 results

Experiment	**Old version**	**New version**
Statoil	33/60 p (55%)	52/60 p (87%)
Students	93/135 p (69%)	122/135% (90%)

The next critical question is number 6 (the question had three alternatives):

6. What is special about the activity "categorize, classify and decide causes"?

2 of 4 answered incorrectly when looking at the old model, while everyone answered correctly when looking at the new. This might be due to that the sub-process symbol used in the original model does not correspond exactly to the one defined in the standard notation overview, as it lacks the "+" a collapsed sub-process is supposed to have attached to it. However, this mismatch is not reflected in the students' responses - all of them answered the question correctly.

Question 9 also got two wrong answers in the original version, and none with the new:

9. The process ends when an accident investigation is carried out

Here, some of the students are also confused: the old version lead to three wrong answers and one unsure (unanswered), whereas the new lead to only correct answers. This question is also event-related, so the reasoning is the same as for question 2.

4 Discussion, Conclusion and Further Work

The quality system of Statoil is developed supporting in particular compliance to requirements to reduce risk, an area where large improvements have been observed over the last decade. Still one find challenges with the comprehension of some of the models as described above. While the requirements given in TR0002 are quite detailed and structured, they are not always followed in practice. Measurements on syntactic quality show that syntax errors are quite common in the workflow models.

The user survey, interviews and conversations provided valuable insights into how users experience the management system. Some measures can be taken to achieve higher quality. The experiment we did gave in a way mixed results; whereas improvement in labelling and syntax appeared to improve the comprehension in one of the cases, the other case which had less severe syntactic errors initially, showed no difference, pointing to that good syntactic quality is useful for comprehension, but that in some cases other aspects are more important if the syntactic quality is sufficiently good.

The main threat to validity in the model quality experiment is that the number of participants was low. Hence, the data is not sufficient for proving or disproving a hypothesis with statistical significance, and the trends discovered may be coincidental. Additionally, students are not part of the target group of the enterprise model, and the findings would have greater validity if all participants were Statoil employees.

Based on the internal evaluation, updated modelling standards and tool support is being developed. When the new functionality has been implemented in full-scale, the actual effect of these changes on model quality in practice can be analysed. A new user survey, similar to the one carried out in 2013/2014 will be distributed by Statoil when these changes have been put into effect. Studying the results based on the new standards and tools and comparing them to the old may give important insight into the real value of such changes. In particular, following the implementation of the new TR0002 document in practice, and how it impacts model quality and use is an interesting possibility for future research. Another possibility is to carry out a more quantitative study, in which an experiment similar to the model quality experiment reported here is carried out in a larger scale with enough respondents to get statistically significant results.

References

1. Heggset, M., Krogstie, J. and Wesenberg. H. Ensuring quality of large scale industrial process collections: Experiences from a case study. In The Practice of Enterprise Modeling, pages 11--25. Springer, (2014).
2. Houy, Constantin, Fettke, Peter, Loos, Peter, van der Aalst, Wil M. P., & Krogstie, John. (2011a). Business Process Management in the Large. *Business & Information Systems Engineering*(6).
3. Krogstie, J.: Model-based development and evolution of information systems: A quality approach, Springer, London (2012)
4. Krogstie, J., Dalberg, V., Jensen, S.M.: Process modeling value framework. In: Manolopoulos, Y., Filipe, J., Constantopoulos, P., Cordeiro, J. (eds.) Selected papers from 8th International Conference, ICEIS 2006, pp. 309–321. Springer, Paphos (2008)
5. Moody DL, Sindre G, Brasethvik T, Sølvberg A (2002) Evaluating the quality of process models: empirical analysis of a quality framework, In Proc. 21st International Conference on Conceptual Modeling (ER'2002), Tampere, Finland, 7-11 Oct
6. Moody, D.L.: Theorethical and practical issues in evaluating the quality of conceptual models: Current state and future directions. Data and Knowledge Engineering 55 243-276 (2005)
7. Nelson, H.J, Poels, G., Genero, M., Piattini, M. A conceptual modeling quality framework. Software Quality Journal 20:201-228 (2012)
8. Price, R.,Shanks, G.: A semiotic information quality framework: Development and comparative analysis. Journal of Information Technology, 20 (2), 88-102 (2005)
9. Silver, B. BPMN Method and Style. Cody-Cassidy Press (2012)
10. Statoil: TR0002 Enterprise Structure and Standard Notation. version 1 (2009)
11. Statoil: TR0002 Enterprise Structure and Standard Notation. version 3 (2013)
12. Statoil: Statoilboken http://www.statoil.com/no/About/TheStatoilBook/Downloads/Statoil-Boken.pdf (2014)
13. Wesenberg, H. Enterprise Modeling in an Agile World PoEM 2011, Proceedings of the 4th conference on Practice of Enterprise Modeling, Oslo, Norway, November 2-3 (2011)

KPI-based Activity Planning for People Working in Flexible Processes

Maikel L. van Eck [*], Natalia Sidorova, and Wil M.P. van der Aalst

Eindhoven University of Technology, The Netherlands
{m.l.v.eck,n.sidorova,w.m.p.v.d.aalst}@tue.nl

Abstract. Planning human activities within business processes often happens based on the same methods and algorithms as are used in the area of manufacturing systems. However, human resources are more complex than machines. Their performance depends on a number of factors, including stress, personal preferences, etc. In this paper we describe an approach for planning activities of people that takes into account business rules and optimises the schedule with respect to one or more KPIs. Taking a task list, a set of rules or constraints and a KPI calculation model as input, we automatically create an executable model that captures all the possible scheduling scenarios. The state space of this executable model is explored to find an optimal schedule.

1 Introduction

Scheduling is a well-known type of optimisation problem that occurs in many different contexts, e.g. flexible manufacturing systems, transportation and personnel planning [2, 4, 9]. Scheduling problems exist in many variations, but often a number of jobs or activities have to be performed by a set of resources while obeying certain constraints. The goal is to divide the activities over the resources and time so that their execution optimises one or more performance measures. In general, scheduling is complex and NP-hard. Different approaches have been developed to deal with this complexity, providing both exact optimal solutions and heuristic approximations.

An aspect that is often not taken into account when planning human activities is that, unlike machines, human performance changes depending on a number of factors [4]. For example, when a person has too much stress, their performance is decreased [5]. By changing for example the lunch time, the performance might be improved or deteriorated. Planning too many challenging activities after each other can also influence the performance.

It has been shown that there are strong relations between work-related stress and deterioration of productivity, absenteeism and employee turnover [3, 5]. Stress and the associated health issues are a big financial burden for both organisations and society in general, with the European Commission estimating the total yearly financial cost at €25 billion. Fortunately, new technological advances

[*] This research was performed in the context of the TU/e IMPULS project.

Activity	CaseID	Time	Employee	Stress
Problem Intake	1	09:00-10:00	John	2
Problem Intake	2	10:00-11:00	John	3
Repair Product	1	11:00-12:00	John	4
Repair Product	2	12:00-13:00	John	6
Break	-	13:00-14:00	John	5
Document Issue	1	14:00-15:00	John	6
Document Issue	2	15:00-16:00	John	8

(a) John's workday of 7 hours, with a maximal stress level of 8.

Activity	CaseID	Time	Employee	Stress
Problem Intake	3	09:00-10:00	Anna	2
Repair Product	3	10:00-11:00	Anna	3
Document Issue	3	11:00-12:00	Anna	4
Break	-	12:00-13:00	Anna	2
Problem Intake	4	13:00-14:00	Anna	3
Document Issue	4	14:00-15:00	Anna	4
Break	-	15:00-15:30	Anna	3
Repair Product	4	15:30-16:30	Anna	4

(b) Anna's workday of 7.5 hours, with a maximal stress level of 4.

Fig. 1: A scheduled workday for two employees, each with a personal schedule.

and smart sensor technologies enable people to unobtrusively monitor their personal stress levels and become more aware of sources of stress at work and stress patterns [8]. In this paper, we explore how the planning of daily activities can be performed with the goal to better manage stress for the employees involved.

As an example, Fig. 1 shows a scheduled workday for two employees working in a hardware maintenance process, whose stress is being monitored. Both employees have each been assigned to work on two cases, but the activities have been scheduled differently. John's schedule in Fig. 1a results in a shorter workday, while Anna's schedule in Fig. 1b results in less stress. Which personal schedule is better depends on the desired tradeoff between time and stress. Of course, due to the effect of stress on performance, the last two activities in John's schedule may actually take longer than usual, so this information also has to be taken into account in the planning.

In this paper we present an activity planning approach to find optimal schedules with respect to one or more *key performance indicators* (KPIs), e.g. stress or workday length. We focus on flexible environments, where planning can be adjusted to people's needs.

The structure of the rest of this paper is as follows: In Sect. 2 we explain our planning approach. Then we discuss our first implementation in Sect. 3 and evaluate it in Sect. 4. Finally, in Sect. 5 we conclude the paper and state several areas for future work.

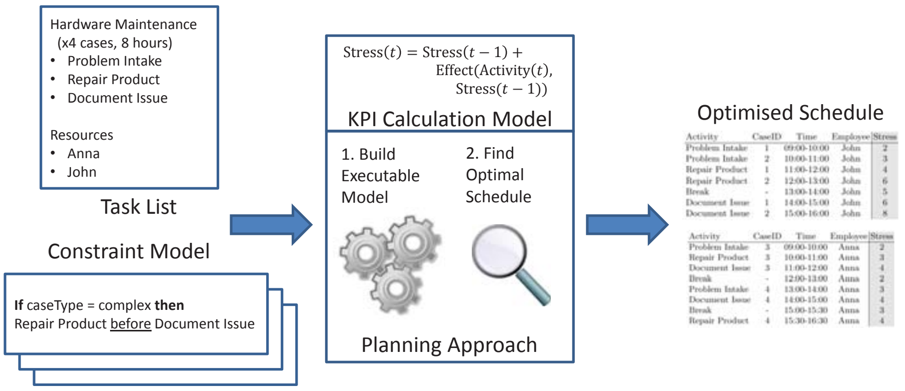
2 Conceptual Approach

A graphical overview of our approach is shown in Fig. 2. We first discuss the input required and then we describe the approach itself.

2.1 Required inputs

We take a task list, the time period in which the activities should take place, a constraint model and a KPI calculation model as input.

The *task list* tells us what activities should be scheduled together with the available resources to divide the activities over. Instead of listing planned activities, the task list can also be extracted from a historical event log. This can be



Activity	CaseID	Time	Employee	Stress
Problem Intake	1	09:00-10:00	John	2
Problem Intake	2	10:00-11:00	John	3
Repair Product	1	11:00-12:00	John	4
Repair Product	2	12:00-13:00	John	6
Break	-	13:00-14:00	John	5
Document Issue	1	14:00-15:00	John	6
Document Issue	2	15:00-16:00	John	8

Activity	CaseID	Time	Employee	Stress
Problem Intake	3	09:00-10:00	Anna	2
Repair Product	3	10:00-11:00	Anna	3
Document Issue	3	11:00-12:00	Anna	4
Break	-	12:00-13:00	Anna	2
Problem Intake	4	13:00-14:00	Anna	3
Document Issue	4	14:00-15:00	Anna	4
Break	-	15:00-15:30	Anna	3
Repair Product	4	15:30-16:30	Anna	4

Fig. 2: An overview of the activity planning approach.

done to see how to improve on a previous execution according to one or more given KPIs. The task list defines the size of the scheduling problem.

The *constraint model* is a collection of rules that have to be upheld for a schedule to be viable. They may describe the order in which activities are executed, specify which people may do what activities, or impose other restrictions on activity executions, e.g. an activity can only be done at a certain location or time. An example of such a rule is "For complex cases, *Repair Product* has to be done before *Document Issue*". Therefore, the schedule in Fig. 1b is only valid if case 4 is a simple case. Rules can even specify that additional activities, not mentioned in the task list, have to be executed. For example, if two activities from the task list are scheduled to be executed sequentially by the same person but at different locations, then that person will need to *travel* between the locations. To specify which people may do what activities the constraint model can define each employee's skills and the competencies needed for each activity [4]. The constraint model also specifies the durations of activities, which may vary depending on different factors. Parameters such as stress levels may influence these durations and the constraint model can describe how, based on patterns found by mining personal historical logs obtained with smart technologies [1,8].

The KPI calculation model defines the quality metrics for schedules. It specifies how the value of one or more specific KPIs can be calculated for a given (part of a) schedule. Examples of KPIs are the length of a working day of people involved, the maximal stress level, and the stress level at the end of the working day. If more than one KPI is specified then a tradeoff needs to be made. This can be done by giving each KPI an importance weight and normalising the KPI values, or by constructing a Pareto front of schedules and allowing the end-user to make the tradeoff [2].

2.2 Planning Approach

The goal of the planning approach is to take the input described above to produce an optimal schedule, or a set of optimal schedules. This approach should ideally not require the end-user to have scheduling expertise or to provide additional input.

Simply generating all permutations of activities divided over time and resources is not practically feasible because there are too many possibilities, while most result in invalid schedules due to violated constraints. A common scheduling technique is to use integer programming, which explores the solution space without explicitly enumerating all possible schedules. However, setting up an integer program is a complex task, especially because of history-dependent variables such as the location of a person, stress level and stress-dependent performance.

Therefore, we propose a planning approach consisting of two stages. First, an executable model that generates valid schedules is automatically constructed from the input described above. Second, this model is used to find the optimal schedule(s).

This approach is similar to the ones in [7,9] on scheduling in flexible manufacturing systems (FMS) and on compliance checking of interrelated compliance rules. In [9], an FMS is modelled as a Petri net (PN), after which a schedule is generated by analysing the state space of the PN. However, the construction of the PN is done manually and the FMS modelling constructs are not as flexible as the rules in the constraint model described above. In [7], a compliance checking framework is described where compliance rules are specified using a formal language and the resulting constructs are translated to Coloured Petri nets (CPNs) [6]. A collection of such CPNs is then composed into one executable model on which the compliance of a given sequence of activities with the rules is checked. However, time is not explicitly modelled and activities that occur in multiple rules occur multiple times in the composed CPN, so it is not possible to use the composed CPN to generate activity schedules.

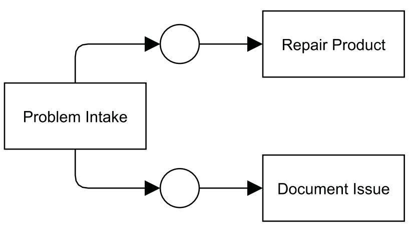
3 Approach Implementation

We have created a preliminary implementation of the proposed activity planning approach as a plug-in in the process mining tool ProM [11]. In this section we discuss how we construct the executable model, specified as a CPN, and how the state space of the executable model is explored to find the optimal schedule.

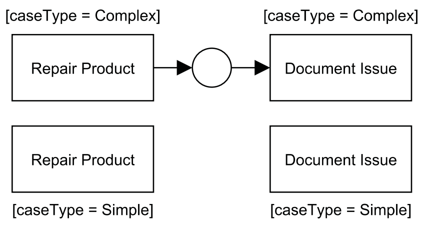
3.1 Building an Executable Model

The executable model is created by combining the inputs described in Sect. 2. It represents the search space of schedules, which is then the input for finding the optimal schedule. To create an executable model, the rules of the constraint model have to be combined and translated to the representation of the executable model.

There are many different ways to express rules or constraints in constraint models [7, 10] We can describe rules in a natural language or use formalisms like LTL. It is also possible to use process models that precisely define what is or is not allowed when executing relevant activities [1]. These models can be



(a) *Problem Intake* occurs before *Repair Product* & *Document Issue*.



(b) If the *caseType* is *Complex*, then *Repair Product* occurs before *Document Issue*.

Fig. 3: Two rules from a constraint model expressed as CPNs.

constructed either by hand or mined from event data related to the involved processes. Another option is to represent each rule as a pattern in a process modelling language, as a mix between creating one big model and describing each rule using a formal rule language.

We choose to model the individual rules of the constraint model as CPNs. One reason for this choice is that CPNs are expressive, but single rules are easier to model than entire processes. Another reason is that CPNs provide us directly with executable models, so we only need to combine them. Two examples of constraints modelled as CPNs are shown in Fig. 3. Activities can be supplied with guards that restrict the conditions relevant to the activity execution, e.g. *caseType* referring to the complexity of a case. When combing rules and creating the schedules we also take into account these guards.

Combining the rules in the constraint model is done using an adapted form of the synchronous product defined for PNs [12]. The adapted synchronous product matches activities by name and then merges not only their dependencies and relations, but also their guards. Guards are merged by taking the conjunction of the guards of the merged activities. Initially, the task list is used to create a basic PN that can execute all required activities exactly as often as needed, which is then sequentially composed with each rule using the synchronous product. The executable model obtained after this step is enhanced with the modelling of resources and time. This results in a single executable model that captures all the rules from the constraint model, as shown by Fig. 4

In our implementation we choose to merge the KPI calculation model with the executable model. Each KPI is tracked separately and their value is updated upon each activity execution. This makes measuring the KPI explicit, even for partial schedules, which helps when finding an optimal schedule. However, the main reason for merging the KPI calculation is that some KPIs, like stress, need to be modelled anyway because they affect the duration of activities.

3.2 Finding an Optimal Schedule

The state space of the executable model described above contains all valid schedules as a final state. The state space is finite, due to the limitation on the number

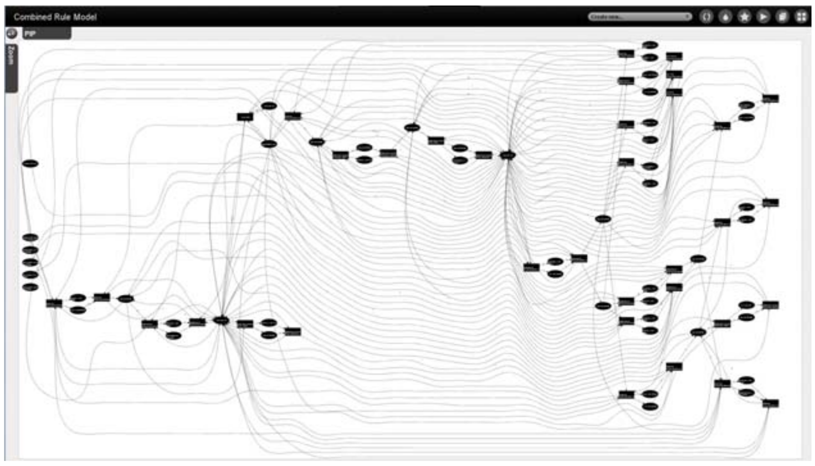


Fig. 4: An automatically constructed executable model in ProM.

of activities that need to be scheduled as well as the bound on the available time to schedule them in. Therefore, one way to find the optimal schedule is to use existing state space exploration techniques to find all final states or deadlocks [9].

Due to the explicit tracking of KPIs and time in each state of the executable model, both constructing a schedule and finding the optimal schedule are straightforward. Given a set of final states, the optimal schedule is the one with the best KPI score or the best tradeoff between KPIs, shown to the user e.g. by creating a Pareto front. Constructing the schedule for a state means traversing the explored state space back to the initial state and recording which activities were executed at what time.

4 Evaluation

We have performed a limited experimental evaluation for the implementation of our approach. A stress monitoring scenario was created, where two employees work at two possible locations on a simple hardware maintenance process. A model explaining the effect of executing activities on the stress of the employees was assumed to be known. Our implementation automatically combined the business rules of the scenario, modelled as CPNs, and scheduled a number of activities, searching for a good tradeoff between stress levels and working time. Two versions of the scenario were tested, one where activity duration did not depend on the stress level and one where stress levels affected performance.

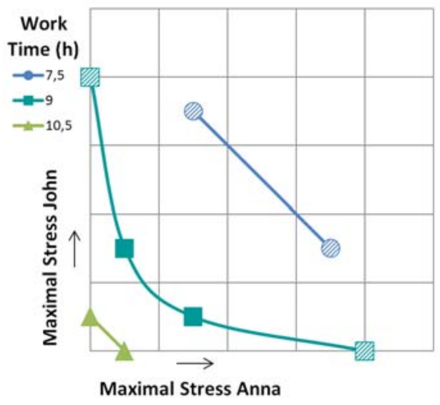
The experimental results are shown in Fig. 5. It is clear from Fig. 5a and Fig. 5b that the number of possible schedules increases exponentially with an increasing number of activities to plan. This also means that the time needed to explore the state space increases exponentially and it quickly reaches a point where the optimal schedule cannot be found within reasonable time. However, it should be noted that the current state space exploration implementation is

Nr. of cases (activities)	Shortest work time	Nr. of valid schedules	Computation time
1 (3)	3 hours	18	0.5 min.
2 (6)	6 hours	160	12.5 min.
3 (9)	7.5 hours	1001	8 hours
4 (12)	?	?	≫24 hours

(a) Planning results with uniform activity durations. The shortest work time is the smallest possible workday length in which all activities can be executed.

Nr. of cases (activities)	Shortest work time	Nr. of valid schedules	Computation time
1 (3)	3 hours	18	0.5 min.
2 (6)	6 hours	130	12 min.
3 (9)	9 hours	396	7 hours
4 (12)	?	?	≫24 hours

(b) Planning results when the effects of stress on performance are considered.



(c) The Pareto front of optimal schedules with 3 cases. The shaded solutions are only valid with uniform activity durations.

Fig. 5: The results of the experimental evaluation.

not very efficient. Another observation is that considering the effects of stress on performance imposes additional restrictions on the solutions, so the state space is smaller and the number of valid schedules is reduced. This also means that the computational complexity of finding the optimal schedule is lower.

Fig. 5c contains a Pareto front of optimal schedules when planning 3 cases. The Pareto front shows that a tradeoff can be made in terms of dividing the work over the available people, as well as the available time. Performing the required work in less time causes higher stress levels. The shaded points indicate schedules that are infeasible if the effect of stress on performance is considered.

5 Conclusion

In this paper we have described an activity planning approach that can find an optimal schedule with respect to one or more KPIs. The approach takes a task list, a set of rules or constraints, and a KPI calculation model to create an executable model that can generate schedules. The state space of this executable model is explored to find an optimal schedule.

Unfortunately, evaluation has shown that the current implementation is not suitable for practical purposes. The implementation allows for a lot of freedom in the types of constraints that can be specified and it can automatically construct an executable model out of these constraints. However, exploring the state space of the resulting executable model is very expensive. Due to the large number of possible ways to schedule activities, the state space becomes too big to explore. There are still multiple areas of future work that can make the suggested approach suitable for practical purposes.

One direction of future work is the use of heuristics during the state space exploration. As the stress of people is highly variable and dependent on many factors, it is difficult to model and predict. Searching for an optimal schedule on a flawed stress model will probably not result in an optimal schedule in practice, so a focus on finding good instead of optimal schedules may be more suitable. The use of state space reduction techniques could also speed up the exploration.

Another direction of future work is the use of a different method to find the optimal schedule. The executable model represents the space of valid schedules, and it might be more efficient to translate this model to a different formalism, e.g. a constraint program. While defining a constraint program directly might be error-prone and challenging, the translation is possible, and would enable the use of many optimisation techniques that exist in constraint programming.

Additionally, more research is needed on learning personalised models that predict people's stress in practical situations and how that affects performance.

References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. Deb, K.: Multi-objective optimization using evolutionary algorithms, vol. 16. John Wiley & Sons (2001)
3. EU-OSHA: Calculating the cost of work-related stress and psychosocial risks (2014), `https://osha.europa.eu/en/publications/literature_reviews/calculating-the-cost-of-work-related-stress-and-psychosocial-risks`
4. Hartmann, S., Briskorn, D.: A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research 207(1), 1–14 (2010)
5. Jamal, M.: Job stress and job performance controversy: An empirical assessment. Organizational behavior and human performance 33(1), 1–21 (1984)
6. Jensen, K.: Coloured petri nets. In: Petri nets: central models and their properties, pp. 248–299. Springer (1987)
7. Jiang, J., Aldewereld, H., Dignum, V., Tan, Y.H.: Compliance checking of organizational interactions. ACM Transactions on Management Information Systems (TMIS) 5(4), 23 (2014)
8. Kocielnik, R., Sidorova, N., Maggi, F.M., Ouwerkerk, M., Westerink, J.H.D.M.: Smart technologies for long-term stress monitoring at work. In: Computer-Based Medical Systems (CBMS), 2013 IEEE 26th International Symposium on. pp. 53–58. IEEE (2013)
9. Lee, D.Y., DiCesare, F.: Scheduling flexible manufacturing systems using petri nets and heuristic search. Robotics and Automation, IEEE Transactions on 10(2), 123–132 (1994)
10. Ramezani, E., Fahland, D., van der Aalst, W.M.P.: Where did I misbehave? Diagnostic information in compliance checking. In: Business Process Management, pp. 262–278. Springer (2012)
11. Verbeek, H.M.W., Buijs, J.C.A.M., Van Dongen, B.F., van der Aalst, W.M.P.: XES, XESame, and ProM 6. In: Information Systems Evolution, pp. 60–75. Springer (2011)
12. Winskel, G.: Petri nets, morphisms and compositionality. In: Rozenberg, G. (ed.) Advances in Petri Nets 1985, Lecture Notes in Computer Science, vol. 222, pp. 453–477. Springer Berlin Heidelberg (1986)

Usability Evaluation of Variability Modeling by means of Common Variability Language

Jorge Echeverria[1], Jaime Font[1], Carlos Cetina[1], and Oscar Pastor[2]

[1] Escuela Politécnica Superior, Universidad San Jorge, Spain,
jecheverria@usj.es, jfont@usj.es, ccetina@usj.es
[2] Centro de Investigación ProS, Universitat Politècnica de València, Valencia, Spain
opastor@dsic.upv.es

Abstract. Common Variability Language (CVL) is a recent proposal for OMG's upcoming Variability Modeling standard. CVL models variability in terms of Model Fragments. Usability is a widely-recognized quality criterion essential to warranty the successful use of tools that put these ideas in practice. Facing the need of evaluating usability of CVL modeling tools, this paper presents a Usability Evaluation of CVL applied to a Modeling Tool for firmware code of Induction Hobs. This evaluation addresses the configuration, scoping and visualization facets. The evaluation involved the end users of the tool whom are engineers of our Induction Hob industrial partner. Effectiveness and efficiency results indicate that model configuration in terms of model fragment substitutions is intuitive enough but both scoping and visualization results suggest that CVL visual notation should be improved.

Keywords: Usability Evaluation, Common Variability Language, Modeling Variability

1 Introduction

Common Variability Language (CVL) has been recently proposed by the architectural board of the OMG as Variability Modeling standard [9]. CVL expresses variability among models in terms of Model Fragments such as Placement Fragments (variation points) and Replacement Fragments (variants). The materialization of product models is performed by means of Fragment Substitutions between a Base Model (Placements) and a Model Library (Replacements).

Usability is a widely-recognized quality criterion essential to warranty the successful use of tools that put the above ideas in practice. This paper presents a usability evaluation of a Modeling Tool augmented with CVL (MT+CVL). The research question addressed by this evaluation is: Are Modeling Tools augmented with CVL intuitive enough to perform the main facets of variability modeling approaches (configuration, scoping and visualization)?

In order to materialize the ideas of CVL, we are going to use our industrial partner Modeling Tool, an induction hobs company that generates their induction hobs' firmwares following a model driven development approach. They used to follow a clone and own approach (without explicit definition of variability) but we have augmented their modeling tool with CVL in order to model the variability existing among their products.

Our Usability Evaluation comprises both (1) test methods such as Performance Measurement, Satisfaction Questionnaire and Interview and (2) inspection methods such as KeyStroke-Level Model [11]. The human computer interaction research community advises to combine these methods to achieve reliable assessment. The selected Usability Evaluation Methods enable us to (1) asses effectiveness, efficiency and satisfaction and (2) to identify usability problems.

Effectiveness and efficiency results (configuration tasks 85% and 132.2%, scoping tasks 65% and 49.93%, visualization tasks 88% and 64.62%) indicate that model configuration in terms of model fragment substitutions is intuitive enough but both scoping and visualization require to improve the visual notation of CVL.

The remainder of this paper is structured as follows: Section 2 discusses related works. Section 3 summarizes the main concepts of the Common Variability Language. Section 4 presents an experimental study to evaluate the usability of the Modeling Tool with CVL. Then, Section 5 describes the results of evaluation and the set of Usability Problems detected. Finally we conclude the paper.

2 Related Work

There are research efforts in literature towards the visualization of SPL (Software Product Line) related artifacts. For instance in [14] the authors present an approach to visualize Pareto-optimal variants (variants, with respect to a set of objectives where no single quality can be improved without sacrificing other qualities). In addition [6] presents an approach that employs visualization and interaction techniques to support end users in the process of product derivation.

There is a concern in existing literature about the comprehensibility of feature models and posible difficulties for different user groups. For instance, in [16] the authors present an experimental approach in understanding of cross-tree constraints in feature models.

In [8] the authors present a Configurable Product Line tool that enable users of the PL to customize it. The authors abstract the technical issues of these customizations to help the users of the PL to understand the implications of decisions made during customization. Furthermore, in [15] the authors are concerned about the flexibility of their product line. Therefore, they present an end-user oriented tool that can support diverse end-users such as project managers, sales people or engineers in their specific tasks.

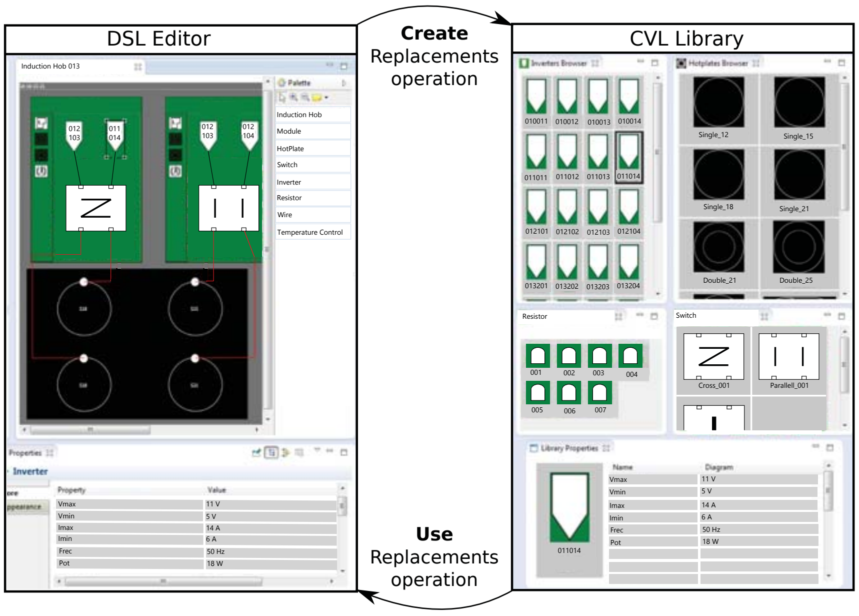
There is also a concern about the usability of DSL and the tool used to generate them. For instance, in [2] the authors present a comparison between five different development tool to create DSLs (and their associated editors). In [5] the authors discuss how user-centered design can be adapted to the context of DSLs development. They argue that usability should be fostered from the beginning of the DSL development cycle, enabling real people to use the DSL.

3 Common Variability Language

CVL is a Domain Specific Language (DSL) for modeling variability in any model of any DSL based on Meta-Object Facility (MOF). The CVL proposal [10, 7] is designed to work in conjunction with an existing DSL editor. Fig. 1 shows an overview of the application of CVL to a given DSL editor. Left part shows the DSL editor itself, while right part represents the library of replacements that will be used to define variants of the base model.

By means of the **use replacement operation**, users can perform substitutions, including fragments from the library into the model being edited. By means of the **create replacement operation**, users can create new replacement fragments and incorporate them into the library [1].

These are the main elements and operations of CVL, and need to be fulfilled to apply CVL for a given DSL. It is necessary to augment the DSL editor in order to enable the operations defined by CVL, but its application is the same for any given DSL. For further details about the inner workings of CVL see [10].



Property	Value
Vmax	11 V
Vmin	5 V
Imax	14 A
Imin	6 A
Frec	50 Hz
Pot	18 W

Name	Diagram
Vmax	11 V
Vmin	5 V
Imax	14 A
Imin	6 A
Frec	50 Hz
Pot	18 W

Fig. 1. Induction Hob MT+CVL

[1] Example of model fragment operations: http://folk.uio.no/oysteinh/demo1.htm

4 Experimental Study

4.1 Context of the Experiment

We have applied CVL to the modeling tool of our industrial partner. That is, we have augmented the Modeling Tool including and integrating the CVL operations and library (as presented in section 3), resulting in the MT+CVL that will be used through the rest of the study. This is the usual operation for augmenting an existing Modeling tool with CVL, and would be the same when applying to any other modeling tool.

Left part of Figure 1 shows the graphical editor of the models. Right part shows the replacements library, where all the replacements that are part of the MT+CVL are shown. In addition, the create replacement operation, enables engineers to create new replacements fragments that are included into the library. The Replace operation enables the engineers to substitute model elements from a product model (open in the editor) by replacements of the replacements library.

4.2 Experimental Object

To evaluate the usability of the MT+CVL we had to know the main tasks that the end users perfom in the main facets of varibility modeling: Scope (in CVL, the creation and elimination of fragments), Configuration (the derivation of products) and Visualization (to make the user aware of the varibility). Six executable tasks were produced as output [2]:

T1 The induction hob IH013 has a problem with the module MOD008 and this module must be replaced by the module MOD014. In the other induction hobs the module MOD008 must not be replaced.
T2 The inverter INV016034 in the module MOD017 in the induction hob IH021 does not run correctly. The module must assemble the inverter INV019034. This replacement must affect every induction hob with the above module.
T3 The induction hob IH021 in the module MOD073 has the inverter INV015034. Its parameter is wrong. A new inverter must be created by cloning the wrong inverter. The new inverter has its parameter VMAX equal to 42. The replacement must affect every induction hob with the above module.
T4 The module MOD021 in IH003 must replace the inverter INV015042 by INV016042. This replacement must not affect to other induction hobs.
T5 To detect all components in the induction hob IH021.
T6 Which is the module most widely used of the set of modules (MOD021, MOD014, MOD017, MOD101)?

The tasks (T1) and (T2) are from configuration facet tasks, (T3) and (T4) are from scope facet tasks and, finally, (T5) and (T6) are from visualization facet tasks.

[2] The identification of the components has been sanitized in order to preserve confidential information. However, omitted information is not relevant for the approach.

4.3 Evaluation Without Users

The Inspection Method (without users) chosen is Keystroke-Level Model. The Keystroke-Level Model has two phases. The first phase is to determine what physical and mental steps a user performs to complete one or more tasks with the CVL Modeling tool in order to predict the time that the user needs to do the task. To do this, a duration is associated to each one of these actions or sequence of operators (physical or mental), and then they are totaled. This duration is calculated by using the average time that it takes a skilled user to complete the action, as suggested by reference time values of [13].

The second phase is to analyze the above steps, looking for problems. Some usability problems that the Keystroke-Level Model might reveal are that it takes too many steps to perform a simple task, or it takes too long to perform the task, or there is too much to learn about the interface, etc. [1]. Furthermore, the amount of time that the user needs to do each task is obtained. In our experiment a Usability Engineer performed every task of section 4.2. For instance, the task1 is composed by four subtasks and the total time predicted to perform the task is 21.1 seconds.

4.4 Evaluation With Users

The objectives of this phase are the assessment on Usability Measures and the identification of usability problems. To achieve these objectives the following UEM are used: Demographic Questionnaire, Performance Measurement, Satisfaction Questionary and Interviews. These UEM are characterized by the participation of the end users. The evaluation with users was as follows [1]:

1. End users were given information about the goals and objectives of the evaluation. They were told that it is not a test of their abilities. They were also informed that their interaction will be recorded.
2. End users attended to a small tutorial about the MT+CVL.
3. End users were asked to fill in a demographic questionnaire.
4. End users were then given a series of clear instructions that were specific for the Performance Measurement. They were advised to try to accomplish the tasks without any assistance, and that they should only ask for help if they felt unable to complete the task on their own.
5. End users were asked to complete the six tasks detailed in the section Experimental Object 4.2. To avoid a possible ceiling effect, there was no time limit to complete the tasks.
6. End users were asked to complete a System Usability Scale questionnaire.
7. End users were asked to answer an interview about CVL Modeling tool.

The evaluation involved the end users of the tool whom are engineers of our Induction Hob industrial partner. The human computer interaction research advises to use five end users in the usability test to obtain 80% of the usability problems [17]. For this reason, we chose a usability evaluation with five end users.

Performance Measurement The goal of this evaluation step was to evaluate how well or poorly the MT+CVL performed for users. Specifically, we measured user effectiveness and efficiency (ISO 1998). An Instructor, an Evaluator and five end users participated in the study. The function of Instructor was to explain the test to the end users and to solve doubts of the end users. The goal of Evaluator was to collect data about the end users action.

In this UEM users performed a predefined set of test tasks (see 4.2) while time and error data was collected. Quantitative data includes performance times, error rates, completed tasks or number of assistance. This data enables the calculation of efficiency and effectiveness. Usability problems will come from the notes that the Evaluator has taken down during the test or extracted from an audio or video recording of the session.

Table 1. Results of Efectiveness and Eficiency

		Unassisted Task Efectiv. (%)	**Assisted Task Efectiv. (%)**	**Time (min)**	**Completion rate/ Task time**	**Asistence**
Configuration	**Mean**	85%	9%	0,86	132,20%	0,2
	Std desv	31%	21%	0,66	55,47%	0,42
	Min	20%	0%	0,16	44,78%	0
	Max	100%	0%	2,23	206,25%	1
Scope	**Mean**	65%	6%	1,91	49,93%	0,1
	Std desv	41%	19%	1,39	41,52%	0,32
	Min	0%	0%	0,57	0,00%	0
	Max	100%	61%	4,80	139,53%	1
Visualization	**Mean**	88%	0%	1,94	64,62%	0
	Std desv	25%	0%	1,75	34,85%	0,00
	Min	36%	0%	0,82	14,96%	0
	Max	100%	0%	6,68	111,11%	0

Measures of effectiveness take into account percent of right finished unassisted tasks, percent of right assisted tasks, frequency of assists to the participant. The efficiency value is the ratio between percent of right finished unassisted tasks and the time to finish these tasks according to Common Industry Format (CIF) for Usability Test Reports [3].

The values in Table 1 indicate that the most difficult or problematic tasks are the scope tasks. In contrast, the end users performed with great easy configuration tasks. On the oher hand, the end users performed correctly the visualization tasks, but it took them too much time taking into account the calculated values with Keystroke-Level Model (see Section 4.3).

Satisfaction Questionary After the performance measurement, a satisfaction questionary was filled by the end users. This questionnaire was System Usability Scale (SUS). SUS was used to determine user's subjective satisfaction with the

SPL tool. Measuring user satisfaction provides a subjective usability metric. The questionnaire was composed by a ten questions with a Likert scale.

The data collected with SUS must be introduced in a spreadsheet to process them. The global score was 73%, which shows that the end users classified the CVL Modeling tool as “good”, according to the scale suggested by [4].

Interview The last UEM used in this phase is Interview. The objectives of this interview were (1) to determine the understanding by the end user of the CVL Modeling tool and (2) to obtain qualitative data from user comments.

The Interview Questions to perform this step had open questions and closed questions. The closed questions were directed to check the understanding of the tasks in the MT+CVL by the end users. For instance, the Instructor shown two pictures to the end user with the state of the CVL Modeling tool after a task and the end user had to choose wich picture is the correct. The open questions aim was to detect the parts of the MT+CVL that were more problematic from a usability point of view, along with the real causes of the problems [12]. For instance, a question was “What have been the more difficult of the tasks for you?”.

5 Conclusion

We believe the results of the usability evaluation are relevant to model-based software developers, OMG variability standardization process and variability tools vendors as follows:

From the point of view of model-based software developers, as the case of our industrial partner, the usability evaluation results suggest that CVL can complement their current modelling tools to formalise and configure variability (according to the result of Effectiveness and Efficiency of variability tasks). The CVL library of model fragments turns out to enable them to shift from a Clone & Own approach to a systematic reuse of model fragments.

From the point of view of current OMGs variability standardization process this paper provides evidence that the current CVL proposal should be extended to provide a visual notation for the model fragment concepts. That is, current CVL proposal introduces the concepts of model placement and model replacement but the proposal lacks a concrete syntax to denote the model fragment boundaries. This lack of visual notation leads modellers to miss variation points in the models.

Finally, from the point of view of tool vendors, the usability evaluation results reveal that modellers require new editing capabilities to work with independent model fragments such as explicit creation, fragment comparison, fragment-based filters and propagations of changes.

References

1. S. Abrahão, E. Iborra, and J. Vanderdonckt. Usability evaluation of user interfaces generated with a model-driven architecture tool. In *Maturing Usability*, pages 3–32. 2008.
2. D. Amyot, H. Farah, and J.-F. Roy. Evaluation of development tools for domain-specific modeling languages. In R. Gotzhein and R. Reed, editors, *System Analysis and Modeling: Language Profiles*, volume 4320 of *Lecture Notes in Computer Science*, pages 183–197. Springer Berlin Heidelberg, 2006.
3. ANSI/NCITS. Ansi/ncits-354 common industry format (cif) for usability test reports. Technical report, NIST Industry USability Reporting, 2001.
4. A. Bangor, P. Kortum, and J. Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *JUS*, 4(3):114–123, 2009.
5. A. Bariic, V. Amaral, and M. Goulao. Usability evaluation of domain-specific languages. In *Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the*, pages 342–347, Sept 2012.
6. G. Botterweck, S. Thiel, D. Nestor, S. bin Abid, and C. Cawley. Visual tool support for configuring and understanding software product lines. In *Software Product Line Conference, 2008. SPLC '08. 12th International*, pages 77–86, Sept 2008.
7. F. Fleurey, Ø. Haugen, B. Møller-Pedersen, G. K. Olsen, A. Svendsen, and X. Zhang. A generic language and tool for variability modeling. *Technical Report SINTEF A13505*, 2009.
8. P. Grunbacher, R. Rabiser, and D. Dhungana. Product line tools are product lines too: Lessons learned from developing a tool suite. In *Automated Software Engineering. 23rd IEEE/ACM International Conference on*, pages 351–354, 2008.
9. Ø. Haugen. Common variability language (CVL) - OMG revised submission. *OMG document ad/2012-08-05*, 2012.
10. Ø. Haugen, B. Møller-Pedersen, J. Oldevik, G. K. Olsen, and A. Svendsen. Adding standardized variability to domain specific languages. In *Proceedings of the 2008 12th International Software Product Line Conference*, SPLC '08, pages 139–148, Washington, DC, USA, Sept 2008. IEEE Computer Society.
11. A. Holzinger. Usability engineering methods for software developers. *Communications of the ACM*, 48(1):71–74, 2005.
12. N. J. Juzgado and X. Ferré. How to integrate usability into the software development process. In *ICSE*, pages 1079–1080, 2006.
13. D. Kieras. Using the keystroke-level model to estimate execution times. *University of Michigan*, 2001.
14. A. Murashkin, M. Antkiewicz, D. Rayside, and K. Czarnecki. Visualization and exploration of optimal variants in product line engineering. In *Proceedings of the 17th Software Product Line Conference*, SPLC '13, pages 111–115. ACM, 2013.
15. R. Rabiser, D. Dhungana, W. Heider, and P. Grunbacher. Flexibility and end-user support in model-based product line tools. In *Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference*, pages 508–511, 2009.
16. I. Reinhartz-Berger, K. Figl, and y. Haugen. Comprehending feature models expressed in cvl. In *Model-Driven Engineering Languages and Systems*, volume 8767 of *Lecture Notes in Computer Science*, pages 501–517. 2014.
17. R. A. Virzi. Refining the test phase of usability evaluation: how many subjects is enough? *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 34(4):457–468, 1992.

Improving Design Patterns Finder Precision Using a Model Checking Approach

Mario L. Bernardi, Marta Cimitile, Giuseppe De Ruvo, Giuseppe A. Di Lucca, and Antonella Santone

Department of Engineering, University of Sannio, Benevento, Italy
e-mail: {mlbernar,dilucca,gderuvo,santone}@unisannio.it
Unitelma Sapienza University, Rome, Italy
e-mail: marta.cimitile@unitelma.it

Abstract. In this paper we propose an approach exploiting the model checking technique to automatically refine the results produced by a Design Patterns mining tool called Design Pattern Finder (DPF) to improve the precision of its results by verifying the detected DPs automatically. To assess the feasibility of the proposed approach along with its effectiveness, we have applied it to an open source Object Oriented system with good results in improving the precision of the detected DPs.

Key words: Software Engineering, Design Patterns, Model Checking, Formal Methods, Models, Mining

1 Introduction

The detection of Design Patterns (DPs) [1] istances in Object Oriented (OO) software systems is valuable to assess the quality of the source code [2] , improve program comprehension, maintenance and reuse [3]. According to this, an increasing interest is adressed to the study and experimentation of DPs detection approaches [4]. Bernardi et al. in [5] have proposed an approach called Design Pattern Finder (DPF), based on a meta-model and a Domain Specific Language (DSL) to represent both the software system and the searched DPs. The DPs models are organized as a hierarchy of declarative specifications and expressed as a wide set of high level properties that can be added, removed or relaxed obtaining new pattern variants. The DPF effectiveness, was evaluated by applying it to several systems and the obtained results are reported in [5]. Even if the obtained results are very encouraging, we observed that the precision of the DPF can be further improved. Indeed, DPF, as any other existing DPs detecting approach, can suffer in lacking of precision and completeness. Starting from these considerations, in this work we exploit formal methods to automatically refine the results produced by DPF; in particular we employ model checking using the Language of Temporal Ordering Specification (LOTOS) and selective-μ-calculus.

The model checking (MC) methodology aims to analyse the number of DPs instances, detected by the DPF, evaluating their correctness with respect to formally encoded properties checked against the entire system model represented with (basic) LOTOS. This allows to reduce the number of wrongly detected patterns (false positives) with respect to the original approach. We decided to apply the MC refinement to the DPF, mainly because DPF is based on a meta-model that can be exploited by the model checking refinement to create (basic) LOTOS processes. Therefore, we embodied a new refinement stage adopting DPF outcomes as inputs. From the DPF model we create (basic) LOTOS processes and from DPF detected patterns we generate selective-μ-calculus properties in order to verify the existence of design patterns through model checking.

The approach has been assessed by a preliminary experiment where it was applied to a system from an open benchmark proposed in [6], [7]. Of course, the proposed refining approach can be extended to any other DP mining approach. The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 gives definitions of basic LOTOS and selective-μ-calculus. Section 4 presents and discusses the proposed detection process, the implemented tools and their integration aspects. Finally, in Section 5, conclusive remarks and future works are presented.

2 Related work

Several pattern recovery techniques and tools have been introduced in the last years. Some reviews about the existing approaches are reported in [4]. Here for brevity, we limit our discussion only to the formal methods (model checking) based approaches. In [8] a formal framework to specify the DPs at different levels of abstraction is proposed. The framework uses stepwise refinement to incrementally add details to a specification after starting from the most abstract one. Moreover, a validation through model checking will verify that a specification in a given level of abstraction is indeed a refinement of a specification of a higher level. The limit of this approach is that a domain specific language to describe DPs is missing and applications in real systems has been never performed. In [9] authors propose an approach aiming to validate DPs using formal method, but the approach is not validated on real software systems. Finally, in [10], a fully automated DPs mining approach performing both static and dynamic analysis to verify the behavior of pattern instances, is proposed. The static analysis exploits model checking to analyze the interactions among objects, while the dynamic analysis of the pattern behavior is performed through a code instrumentation and monitoring phase, applied on the candidate pattern instances. This approach, differently from ours, requires the analysis of the collaboration among objects at runtime by identifying and executing test cases on the software system.

3 Preliminaries

Let us now recall the main concepts of Basic LOTOS [11]. A Basic LOTOS program is defined as:

```
process ProcName := B
where E
endproc
```

where `B` is a *behaviour expression*, `process ProcName := B` is a *process declaration* and `E` is a *process environment*, i.e., a set of process declarations. A behaviour expression is the composition, by means of a set of operators, of a finite set $\mathcal{A}$=$\{$`i,a,b, ...`$\}$ of atomic *actions*. Each occurrence of an action in $\mathcal{A}$ represents an event of the system. An occurrence of an action `a` $\in \mathcal{A}-\{$`i`$\}$ represents a communication on the gate `a`. The action `i` does not correspond to a communication and it is called the *unobservable action*. The syntax of behaviour expressions (also called *processes*) is the following:

```
B ::= stop | α;B | B[]B| P | B|[S]|B | B[f] | hide S in B | exit
                              | B>>B | B[>B
```

where `P` ranges over a set of process names and α ranges over $\mathcal{A}$. he following:

- The *action prefix* `a;B` means that the corresponding process executes the action `a` and then behaves as `B`.
- The *choice* `B1 [] B2` composes the two alternative behavior descriptions `B1` and `B2`.
- The expression `stop` cannot perform any move.
- The *parallel composition* `B1|[S]|B2`, where `S` is a subset of $\mathcal{A}-\{$`i`$\}$, composes in parallel the two behaviors `B1` and `B2`. `B1` and `B2` interleave the actions not belonging to `S`, while they must synchronize at each gate in `S`. A synchronization at gate `a` is the simultaneous execution of an action `a` by both partners and produces the single event `a`. If `S`$=\emptyset$ or `S`$=\mathcal{A}$, the parallel composition means pure interleaving or complete synchronization.
- Cyclic behaviors are expressed by recursive process declarations.
- The *relabeling* `B[f]`, where `f`: $\mathcal{A} \rightarrow \mathcal{A}$ is an action relabeling function, renames the actions occurring in the transition system of `B` as specified by the function `f`. This function is syntactically defined as `a0 -> b0,...,an->bn`, meaning `f(a0)=b0,...,f(an)=bn`, and `f(a)=a` for each `a` not belonging to $\{$`a0,...,an`$\}$. Note that each relabelling function has the property that `f(i)` = `i`.
- The *hiding* `hide S in B` renames the actions in `S`, occurring in the transition system of `B`, with the unobservable action `i`.
- The expression `exit` represents successful termination; it can be used by the enabling (`B >> B`) and disabling (`B[> B`) operators: `B >> B` represents sequentialization between `B1` and `B2` and `B[> B` models interruptions. For the sake of simplicity, we do not discuss these operators in the paper.

The semantics of a process `B` is rules describing the transition relation of the automaton corresponding to the behavior expression defining `B`. This automaton is called *standard transition system* for `B` and is denoted by $\mathcal{S}$(`B`). The reader can refer to [11] for details. From now on, we write LOTOS instead of Basic LOTOS.

In the following we recall the selective-μ-calculus, introduced in [12], which is a branching temporal logic to express behavioral properties of systems. It is equi-expressive to μ-calculus [13], but it differs from it in the definition of the modal operators. Given a set $\mathcal{A}$ of actions and a set Var of variables, the selective-μ-calculus logic is the set of formulae given by the following inductive definition:

- `tt` and `ff` are selective-μ-calculus formulae;
- Y, for all $Y \in Var$, is a selective-μ-calculus formula;
- if φ_1 and φ_2 are selective-μ-calculus formulae then $\varphi_1 \wedge \varphi_2$ or $\varphi_1 \vee \varphi_2$ are selective-μ-calculus formulae;
- if φ is a selective-μ-calculus formula then $\langle K \rangle_R\, \varphi$ and $[K]_R\, \varphi$ are selective-μ-calculus formulae, where $K, R \subseteq \mathcal{A}$;
- if φ is a selective-μ-calculus formula then $\mu X.\varphi$ and $\nu X.\varphi$ are selective-μ-calculus formulae, where $X \in Var$.

The satisfaction of a formula φ by a state s of a transition system, written $s \models \varphi$, is defined as follows: each state satisfies `tt` and no state satisfies `ff`; a state satisfies $\varphi_1 \vee \varphi_2$ ($\varphi_1 \wedge \varphi_2$) if it satisfies φ_1 or (and) φ_2. $[K]_R\, \varphi$ is satisfied by a state which, for every performance of a sequence of actions not belonging to $R \cup K$, followed by an action in K, evolves to a state obeying φ. $\langle K \rangle_R\, \varphi$ is satisfied by a state which can evolve to a state obeying φ by performing a sequence of actions not belonging to $R \cup K$, followed by an action in K. The precise definition of the satisfaction of a closed formula φ by a state of a transition system can be found in [12].

4 Approach

The overall Design Pattern mining approach follows a process structured in two main sub-processes. The first performs the design pattern detection applying the Graph-Matching approach implemented by DPF [5]. The second performs the refinement of DPF results using the model checking approach proposed in this paper.

In the following is a short description of each process activity, while next sub-sections will provide more details about them:

- **Source Code Analysis** — The source and bytecodes of the system under study are parsed and the complete ASTs of the system are produced.
- **Model Instantiation** — A traversal of the system AST is performed to generate an instance of the system model (i.e. the system graph S), conforming to the meta-model defined for DPF. Rapid type analysis (RTA), class flattening and inlining of not public methods are exploited in order to build a system's representation suitable for the matching algorithm.
- **Graph-Matching DPs Detection** — The DPF graph matching algorithm, described in [5], is performed to match the system graph, built in the previous step, with the pattern specifications graphs of the DPs to be detected.

- **Pattern2MU** — Each pattern specification to be detected is written as a set of templates μ-properties (also MU-properties used in the following). These properties involve the patterns roles and their relationships. The template parameters are bound to the concrete system elements using information extracted from the pattern instances found in the detection step (i.e. roles and the system elements related to them).
- **Model2LOTOS** — In order to check if a given set of parametrized MU-properties holds, the system graph should be expressed in a suitable model (in our approach LOTOS was exploited). Hence this step takes the system graph as input and translates it to a LOTOS model instance. This translation has to be performed only one time for each system to be mined.
- **Results refinement** — This step checks the parametrized sets of MU-properties obtained from the pattern specifications catalogue against the LOTOS model of the system in order to reduce the number of false positives.

4.1 Graph-Matching DPs Detection

The detection of the DPs instances is performed according to the DPF approach [5], based on a meta-model and a Domain Specific Language (DSL) to model the structure of both OO systems and DPs. Each pattern, in order to be detected, is modeled by a DSL pattern specification that can be translated into DP Graph (DPG) which is part of the input for the graph-matching detection algorithm.

Along the execution of the DPF Graph Matching algorithm, the system graph (i.e., the instance of the system model) is traversed and each pattern instance sub-graph is mapped to the corresponding matching DPG (to identify the actually implemented patterns). More insights and details about the DPF approach can be found in [5].

4.2 DPF Refinement

The proposed approach is based on the use of formal methods (to the authors' knowledge, never used before). From the DPF outcomes we derive LOTOS processes, which are successively used to perform model checking. The goal of the approach is to increase the precision of DPs mining results produced by DPF. This part of the approach is addressed by the second sub-process which comprises the following steps:

1. LOTOS System model creation (Model2LOTOS activity)
2. Pattern Property generation (Pattern2MU activity)
3. Pattern Matching through Model Checking (Results Refinement activity)

In the following subsections the three steps are discussed in detail.

LOTOS model creation We use, as internal representation, the LOTOS language. Thus, LOTOS specifications are generated starting from the internal representation of DPF. This is obtained by defining a DPF-to-LOTOS transform operator $\mathcal{T}$. The function $\mathcal{T}$ directly applies to Java system outcomes of DPF and translates them into LOTOS process specifications. The function $\mathcal{T}$ is defined for each part of a Java system such as classes, interfaces, methods, fields. Each one has been translated into LOTOS processes. First of all, a System is composed of a set of Types. A Type may be a ClassType or an InterfaceType. A ClassType is made up of Methods. Types may be tied by inheritance relations and a ClassType may implement an InterfaceType, as usually occurs in OO software systems.

System
The generic Java *System* containing k types is translated into the following LOTOS process:

$$\mathcal{T}(C) = \textit{ process } SYSTEM := Type_1[] \cdots []Type_k \textit{ endproc}$$

where $Type_i$ is written using the fully qualified Java name. The LOTOS process $SYSTEM$ represents the parent process of all the types. Each translated LOTOS model has a *System* process.

Type
As stated, a Type may be a ClassType or an InterfaceType. For example, if FQN is the fully qualified name of a Type, an InterfaceType is translated into the following LOTOS process:

$\mathcal{T}(I) = process$
$FQN\_InterfaceType :=$
$name\_InterfaceType; (FQN\_\ Method_i; FQN\_\ Method_i\_\ Method[] \cdots []$
$FQN\_\ Method_k; FQN\_\ Method_k\_\ Method[]$
$inherits; (FQN\_\ InterfaceType_l[] \cdots []$
$FQN\_\ InterfaceType_y))$
$endproc$

where *implements* and *inherits* are actions which respectively indicate implementation of interfaces and inheritance relation between types.

Method
A method is represented with its own arguments and with a modifier, thus it is translated into the following LOTOS process:

$\mathcal{T}(M) = process$
$FQN\_\ Method := name\_Method; (arg_i[] \cdots []arg_k[]modifier\_mod)$
$endproc$

where arg_i is the name of the argument and mod is the type of modifier such as public, private, protected.

Pattern Property generation In our approach, we use model checking to verify the existence of specific patterns. Once we have the LOTOS processes of the Java software system, we can use selective-μ-calculus logic to specify desired properties. A pattern is translated into a selective-μ-calculus property. Each design pattern leads to a different property, although a set of common properties are used as building blocks:

1. Existence of Interface Implementation:
   $\langle implements \rangle_\emptyset \, \langle name\_\, InterfaceType \rangle_\emptyset \, \mathtt{tt}$
2. Existence of Inheritance:
   $\langle inherits \rangle_\emptyset \, \langle name\_\, ClassType \rangle_\emptyset \, \mathtt{tt} \wedge \langle inherits \rangle_\emptyset \, \langle name\_\, InterfaceType \rangle_\emptyset \, \mathtt{tt}$
3. Existence of a Method:
   $\langle name\_\, Method \rangle_\emptyset \, \mathtt{tt}$
4. Existence of a Field:
   $\langle field \rangle_\emptyset \, \langle name\_InterfaceType \rangle_\emptyset \, \mathtt{tt} \wedge \langle field \rangle_\emptyset \, \langle name\_ClassType \rangle_\emptyset \, \mathtt{tt}$
5. Existence of an Argument:
   $\langle arg \rangle_\emptyset \, \langle name\_\, InterfaceType \rangle_\emptyset \, \mathtt{tt} \wedge \langle arg \rangle_\emptyset \, \langle name\_\, ClassType \rangle_\emptyset \, \mathtt{tt}$

Pattern Matching through Model Checking Once we have created the LOTOS model of a Java software system and we also have built all the properties which represent the design patterns, we can proceed with model checking. As aforementioned, in this paper both model and properties (patterns) come out translating the ones of DPF. We have used CADP [14] as formal verification environment. The CADP model checker is applied verifying each pattern against the System model. When the result is TRUE, it means that the pattern has been found. FALSE otherwise. Thanks to a very detailed LOTOS model we are able to detect false positives of DPF.

5 Conclusions and future works

In this work we exploit formal methods to automatically refine the results produced by a previous approach called DPF. DPF approach introduces a meta-model to represent both the patterns and the system under study as graphs in order to apply a graph matching algorithm. In this paper the detection process is enriched with a model-checking refinement step in which the system model is represented using LOTOS and patterns as selective-μ-calculus properties checked against it. The defined LOTOS model allows to check a wider set of properties that lead to a reduction of the number of false positives. Preliminary experiments performed on a middle sized system (QuickUML 2.1) confirmed the feasibility, correctness, and effectiveness of the approach showing, improvement of the precision (30% on average) with a very reduced impact on the original recall. The model-checking step indeed reduced to zero the number of false positives for Command and Strategy patterns, raising the precision, respectively, from 0.88 and 0.67 to 1. In QuickUML system in both cases the MC properties were able to consider structural or behavioral relationships that the original DPF approach was unable to take into account.

As future works, a more complete translation of pattern specifications to selective-μ-calculus properties will be defined. Moreover, we want to develop new user friendly tools to assist software engineers during the model checking step, as done in [15]. Finally, we plan to perform the translation of the entire DP catalogue defined in [5] as selective-μ-calculus properties.

References

1. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1995)
2. Bergenti, F., Poggi, A.: Improving uml designs using automatic design pattern detection. In: SEKE 2000. (2000) 336–343
3. L. Prechelt, B. Unger-Lamprecht, M.P., Tichy, W.: Two controlled experiments assessing the usefulness of design pattern documentation in program maintenance. IEEE Trans. Softw. Eng. **28**(6) (2002) 595–606
4. Rasool, G., Streitfdert, D.: A survey on design pattern recovery techniques. IJCSI International Journal of Computer Science Issues **8**(2) (2011) 251 – 260
5. Bernardi, M., Cimitile, M., Di Lucca, G.: Design patterns detection using a dsl-driven graph matching approach. Journal of Software: Evolution and Process **Wiley Online Library** (2014)
6. Guéhéneuc, Y.G.: P-mart: Pattern-like micro architecture repository,. In: Proceedings of the 1st EuroPLoP Focus Group on Pattern Repositories, Michael , Aliaksandr Birukou, and Paolo Giorgini (2007, `http://www.ptidej.net/tool/designpatterns/`)
7. : Comsats institute of information technology. `http://research.ciitlahore.edu.pk/Groups/SERC/DesignPatterns.aspx`
8. Taibi, T., Herranz-Nieva, Á., Moreno-Navarro, J.J.: Stepwise refinement validation of design patterns formalized in TLA+ using the TLC model checker. Journal of Object Technology **8**(2) (2009) 137–161
9. Aranda, G., Moore, R.: A formal model for verifying compound design patterns. In: Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering. SEKE '02, New York, NY, USA, ACM (2002) 213–214
10. De Lucia, A., Deufemia, V., Gravino, C., Risi, M.: Improving behavioral design pattern detection through model checking. In: CSMR, 2010. (2010) 176–185
11. Bolognesi, T., Brinksma, E.: Introduction to the iso specification language lotos. Computer Networks **14** (1987) 25–59
12. Barbuti, R., De Francesco, N., Santone, A., Vaglini, G.: Selective mu-calculus and formula-based equivalence of transition systems. J. Comput. Syst. Sci. **59**(3) (1999) 537–556
13. Stirling, C.: An introduction to modal and temporal logics for ccs. In: Concurrency: Theory, Language, And Architecture. (1989) 2–20
14. Garavel, H., Lang, F., Mateescu, R., Serwe, W.: CADP 2011: a toolbox for the construction and analysis of distributed processes. STTT **15**(2) (2013) 89–107
15. De Ruvo, G., Santone, A.: An eclipse-based editor to support lotos newcomers. In: WETICE, 2014 IEEE 23rd. (June 2014)

Patterns for Identifying and Structuring Features from Textual Descriptions: An Exploratory Study

Nili Itzik and Iris Reinhartz-Berger

Department of Information Systems, University of Haifa, Israel
nitzik@campus.haifa.ac.il, iris@is.haifa.ac.il

Abstract. Software Product Line Engineering (SPLE) supports developing and managing families of similar software products, termed Software Product Lines (SPLs). An essential SPLE activity is *variability modeling* which aims at representing the differences among the SPL's members. This is commonly done with feature diagrams – graph structures specifying the user visible characteristics of SPL's members and the dependencies among them.
Despite the attention that feature diagrams attract, the identification of features and structuring them into feature diagrams remain challenging. In this study, we utilized Natural Language Processing (NLP) techniques in order to explore different patterns for identifying and structuring features from textual descriptions. Such a catalog of patterns is important for both manually-created and automatically-generated feature diagrams.

Keywords: Variability Analysis, Feature Diagrams, Natural Language Processing, Empirical Evaluation

1 Introduction

Software Product Line Engineering (SPLE) supports developing and managing similar software products (SPLs) [14]. SPLE has been proven to be successful in reduction of development cost, time-to-market and improvement of product's quality [12]. Variability modeling is a crucial activity for identifying and documenting the precise differences among the SPL's members for effective and efficient development and management of the entire SPL. Feature diagrams [3], which are commonly the outcomes of the variability modeling activity, are graph (or tree) structures that describe "features" of a SPL and the relationships and dependencies among them [9]. A "feature" can be defined as "a prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems" [9].

Variability models in general and feature diagrams in particular are created either manually by humans or automatically utilizing methods such as [2], [4], [15], [18]. Being manually or automatically created, the identification and structuring of features are important for the comprehensibility of the models. In this study we explored different semantic patterns to create feature diagrams from textual descriptions. The descriptions were short and focused on the (visible) behaviors of the SPLs' members. We further grounded these patterns utilizing Natural Language Processing (NLP) techniques and proposed guidelines for their use. Thus, the contribution of the work is two-fold. First, the patterns provide guidelines for modelers who create feature diagrams as to how to identify and structure features. Second, the patterns may be the basis for flexible automatic feature extraction processes.

The rest of the paper is structured as follows. Section 2 briefly reviews related work. Section 3 describes the study's settings and execution, while Section 4 presents the patterns extraction process and outcomes. Section 5 discusses the findings and refers to limitations. Finally, Section 6 concludes and suggests future directions.

2 Related Work

As noted, feature diagrams can be created manually or automatically from text. Only a few studies suggest guidelines for (manually) creating feature diagrams. In [10] guidelines for identifying features and classifying them according to the types of information they represent are suggested. The four classes of features are: capability, operating environment, domain technology, and implementation technique. Organization of the features into diagrams is then done by analyzing the relations between these classes of features. In [11] further guidelines have been suggested for domain planning, feature identification, feature organization, and feature refinement. For example, it is recommended not to "organize features to represent functional dependencies," but to "capture and represent commonalities and differences."

Studies that use textual descriptions for automatically (or semi-automatically) identifying and extracting features relay on syntactic patterns, utilizing different parts-of-speech (POS). Examples of such studies are [2], [4], [5], [7], [13], [18]. They mainly use nouns and verbs for this purpose. Some of them suggest structuring the features using different clustering algorithms. In [8], an automatic Semantic and Ontological Variability Analysis (SOVA) method is suggested to construct feature diagrams based on behavioral similarity and variability. The input textual descriptions are parsed according to the semantic roles of the phrases in the sentences and the behavioral elements are extracted utilizing an ontological model. Then the semantic similarity between the behavioral elements is used for creating feature diagrams.

There are also studies that generate features diagrams from feature configurations using refactoring techniques, e.g., [1] and [16]. In these studies the inputs are some structured or formal representations (such as, propositional formulas or feature lists and dependencies). Identification of features is not needed, as the features are already given. Organization of features into diagrams is done by analyzing implication graphs, which are directed graphs with features as vertices and edges that represent dependencies between features.

To summarize, existing studies provide some general guidelines for creating feature diagrams. Others extract features using specific, pre-defined syntactic or grammatical patterns. The organization of features into diagrams in those studies is done using transformation rules or clustering algorithms, without referring to the extracted features characteristics (e.g., their POS). In our study, we explored the various semantic patterns that can represent features, as well as the relations among them.

3 Study's Settings and Execution

In order to explore the various ways modelers extract features from textual descriptions and organize them into feature diagrams, we developed a questionnaire with eight short paragraphs. Each paragraph described a SPL including information on the applications' behaviors and the allowed variability in the SPL. The task was to present for each description a feature diagram that resembles the description[1].

The participants in this study were 11 information systems students at the University of Haifa, Israel. Those students participated in an advanced software engineering course that was devoted to SPLE. The descriptions referred to application domains expected to be familiar to students: e-shop, library management, photo sharing, and text editing. Overall we received 78 feature diagrams from the eleven participants, as a few participants answered the questionnaire partially.

4 Outcomes: Feature Patterns and their Relations

Looking at the obtained feature diagrams, we observed that the participants used various linguistic and semantic parts of the original descriptions when naming the features. Therefore, we decided to utilize NLP techniques to analyze the results. Particularly, we decided to use the Semantic Role Labeling (SRL) technique [6], which refers to the semantic roles of phrases in particular sentences and goes beyond POS. Next we elaborate on SRL, the patterns we identified, and the relations found between the extracted patterns when organizing the features into diagrams.

4.1 Semantic Role Labeling

SRL [6] associates constituents of a phrase with their semantic roles in the phrase. Those semantic roles identify the relationships that a syntactic constituent has with a predicate. Typical semantic arguments include: (1) **Agent (A0)** – Who performs the action?; (2) **Object (A1)** – On what object is the action performed?; (3) **Instrument (A2)** – How is the action performed? Identification of adjunctive arguments, termed **modifiers**, is further supported in SRL, for example: Temporal (AM-TMP) – When is the action performed? or Adverbial (AM-ADV) – In what conditions is it performed?

The benefits of SRL for analyzing variability of functional requirements have already explored in [15][2]. As an example consider the following sentence:

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

[2] We used the English version of SRL. As the text descriptions were given in Hebrew – the mother tongue of the participants, we had to translate them to English. Two researchers verified the translation and especially its accuracy with respect to POS.

The users subscribe to the site and update their profiles, using an online interface.

Two verb predicates are identified in this sentence: 'subscribe' and 'update'. Accordingly, the extracted roles (marked in square brackets) are:

The users$_{\text{[Agent]}}$ subscribe$_{\text{[action]}}$to the site$_{\text{[object]}}$using an online interface$_{\text{[AM-ADV]}}$

The users$_{\text{[Agent]}}$update$_{\text{[action]}}$their profile$_{\text{[object]}}$using an online interface$_{\text{[AM-ADV]}}$

The features extracted from this sentence are expected to include “update profile” and “subscribe to site.” However, these features could appear in different contexts. Fig. 1 demonstrates three such contexts for the feature “update profile”: the agent who performs the action (a), the object on which the action is performed (b), and the action itself (c).

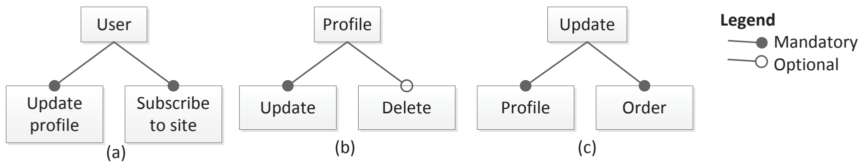


Fig. 1. Possible contexts to the feature “update profile”

4.2 Identified Feature Patterns

To identify the feature patterns we mapped the features that were specified by the participants to the different descriptions to the outcomes of the SRL technique on those descriptions. The extracted patterns are listed in Table 1. Since instruments and modifiers played similar roles in our descriptions – they both describe actions – currently we do not distinguish between patterns based on those roles. Furthermore, the other roles, namely, agents, actions, and objects, appear in (almost) any sentence, while modifiers and instruments interchangeably appear, if at all.

As can be seen the commonly used patterns in our study described partial functionality (i.e., combination of actions and the objects on which they are performed, e.g., “update profile”) and objects (e.g., profile). These are followed by “descriptive” and “actions” patterns, which utilize different modifiers/instruments and the sentences' predicates, respectively. We further observed stakeholders-related patterns, namely, patterns involving the agent role, and different combinations of functionality-related roles, e.g., Action+Modifier/Instrument and Action+Object+Modifier/Instrument.

Table 1. The extracted feature patterns

Pattern Name	Patterns Content	Example (modifier type)	# occurrences
Partial functionality	Action+Object	"make order"	314
Objects	Object	"price"	246
Descriptive	Modifier/ Instrument	"via credit card" (AM-MNR)	230
Actions	Action	"purchase"	163
Stakeholders	Agent	"supplier"	59
Described actions	Action+ Modifier/ Instrument	"view by popularity" (Instrument)	39
Described partial functionality	Action+ Object+ Modifier/ Instrument	"prices automatically updated" (AM-MNR)	9
Described objects	Object+ Modifier/ Instrument	"operations on file" (AM-LOC)	9
Actions by stakeholders	Agent+Action	"users register"	5
Described stakeholders	Agent+ Modifier/ Instrument	"retrieval interface from supplier's site"	4
Full functionality	Agent+ Action+ Object	"Suppliers publish items"	3

4.3 Relations of Feature Patterns

We further tried to examine the relations between the features' patterns when organizing the features into diagrams. To this end, we examined for each one of the five top found patterns what patterns their descendants follow, and particularly their direct child features. Table 2 summarizes those findings.

- A child of a feature following the **"partial functionality"** pattern commonly follows "descriptive", "objects", or "partial functionality" patterns. This means that the child refers to other aspects of the functionality (as in "descriptive" and "objects" patterns) or refines the parent (as in "partial functionality" pattern).
- A child of a feature following the **"objects"** pattern commonly follows "objects" to refine the parent. However, it can also follow "descriptive", "partial functionality", or even "actions" to describe or specify the possible uses of the objects.
- A child of a feature following the **"descriptive"** pattern commonly follows "descriptive", "actions", or "partial functionality" patterns. In many cases the "descriptive" pattern appears in the leaves of the diagram.
- A child of a feature following the **"actions"** pattern follows "descriptive", "objects", "actions", or "partial functionality" patterns.
- A child of a feature following the "**stakeholders**" pattern mainly follows the "partial functionality" pattern that describes the actions which are performed by the stakeholders and the objects on which they are performed. Other children's patterns were also observed in this case, most notably, "actions" and "objects".

Table 2. Relations between parent’s and child‘s patterns

Parent’s pattern → **Child’s pattern ↓**	**Partial functionality**	**Objects**	**Descriptive**	**Actions**	**Stakeholders**
Partial functionality	35	50	22	14	74
Objects	58	62	11	20	26
Descriptive	72	53	26	35	8
Actions	9	18	23	16	25
Stakeholders	5			2	7
Described actions	11	2	1	9	2
Described objects		1	2	1	
Described partial functionality		3	4	2	
Full functionality	2				
Actions by stakeholders	1				

5 Discussion and Limitations

We identify a number of interesting results that worth further discussion. First, we found that in many cases the features describe functionality-related aspects. These findings are in-line with the definitions of features that many of them highlight the functional part of the features.

Second, the top found patterns refer to either functionality or structure. In [8], we have already reported that most feature diagrams in S.P.L.O.T[3] – an academic repository of feature diagrams – followed to some extent a structural or a functional perspective. A structural perspective corresponds to our “objects” and “described objects” patterns. A functional perspective is highly related to our “actions”-involving patterns. We further found that actions were usually accompanied with the objects on which they are performed (corresponding to our “partial functionality” pattern).

Third, our findings can be directly transformed into guidelines for modeling feature diagrams from textual descriptions:

1. Extract actions (potentially with their associated objects), objects, modifiers, and agents from the textual descriptions. Examine whether they can serve as features, namely, their variability is of interest.
2. Try to refine each of the extracted features with the same pattern (used to extract the feature) or with the other top found patterns.
3. Examine the parts of the descriptions not covered by the previous steps. Try to use the other patterns to fully model variability.

Finally, our patterns cover the ways existing methods extract features from textual descriptions. This originates from the fact that our patterns are semantic, as opposed

[3] S.P.L.O.T Software Product Lines Online Tools. http://www.splot-research.org/

to the syntactic and grammatical patterns used by the existing approaches. The same sequence of POS can be mapped to different semantic roles. For example, adjective+noun can be used for describing the agent (e.g., registered user) or the object (e.g., small items). As such, the suggested catalog goes beyond the syntactical structure of the sentence. It can further be used by those methods to enable more flexible generation of feature diagrams and may set the ground for systematic methods to identify and structure SPL features.

The validity of our study is subject to several threats. First, the knowledge and skills of our participants may be questioned. However, they were students in an advanced software engineering course who had the required background in SPLE and feature modeling. The use of students in different software engineering research areas is acceptable as it was shown that students have a good understanding of the way industry behaves [17]. Second, the relatively low number of participants may challenge the ability to generalize the results. Thus, each participant was required to model several feature diagrams, resulting with 78 diagrams overall. Although these diagrams are not independent, it enabled us analyzing more cases. Another threat is the possibility that the way the descriptions were phrased influenced the feature extraction process. Thus, we used eight different descriptions. We did not use a fixed, predefined way to phrase those descriptions. Finally, the questionnaire used in this study was written in Hebrew and translated to English in order to apply the SRL technique. This may affect the pattern extraction process. To overcome this threat, a researcher not involved in the current study additionally verified the translation in general and with respect to POS in particular. Following her feedback, a few corrections were made prior to execution of the study.

6 Summary and Future Work

Variability modeling is an important activity in Software Product Line Engineering (SPLE). Extraction of features and structuring them into diagrams are challenging, time-consuming, and error-prone. In this paper we present a catalog of patterns that can be used to extract features from textual descriptions. These patterns are based on semantic considerations (rather than syntactic and grammatical ones). We further discuss the relations between those patterns in order to assist in organizing the features hierarchically and creating feature diagrams. As far as we know, we are the first ones to create a catalog of semantic patterns and use it to create feature diagrams.

Further work is required to replicate the study with different experienced populations of participants and different textual descriptions (in terms of length and phrasing styles). The usefulness of the patterns for modeling variability and automatically generating feature diagrams needs to be explored as well. Finally, exploration and analysis of the indirect relations between patterns and refinement of patterns based on existing or additional roles may provide interesting findings that can help improve feature diagrams creation processes.

References

1. Acher, M., Baudry, B., Heymans, P., Cleve, A., & Hainaut, J. L. (2013). Support for reverse engineering and maintaining feature models. Workshop on Variability Modelling of Software-intensive Systems (VaMoS), Article #20.
2. Acher, M., Cleve, A., Perrouin, G., Heymans, P., Vanbeneden, C., Collet, P.,& Lahire, P. (2012). On extracting feature models from product descriptions. Workshop on Variability Modelling of Software-intensive Systems (VaMoS), pp. 45-54.
3. Chen, L., & Babar, M.A. (2011). A systematic review of evaluation of variability management approaches in software product lines. Information and Software Technology 53, pp. 344-362.
4. Davril, J. M., Delfosse, E., Hariri, N., Acher, M., Cleland-Huang, J., & Heymans, P. (2013). Feature model extraction from large collections of informal product descriptions. The 9th Joint Meeting on Foundations of Software Engineering, pp. 290-300.
5. Ferrari, A., Spagnolo, G. O., & Dell'Orletta, F. (2013). Mining commonalities and variabilities from natural language documents. Software Product Line Conference (SPLC'13), pp. 116-120.
6. Gildea, D. & Jurafsky, D. (2002). Automatic Labeling of Semantic Roles. Computational Linguistics 28 (3), pp. 245-288.
7. Hariri, N., Castro-Herrera, C., Mirakhorli, M., Cleland-Huang, J., & Mobasher, B. (2013). Supporting domain analysis through mining and recommending features from online product listings. IEEE Transactions on Software Engineering 39(12), pp. 1736-1752.
8. Itzik, N. & Reinhartz-Berger, I. (2014). Generating Feature Models from Requirements: Structural vs. Functional Perspectives. Software Product Line Conference (SPLC'14) – Volume 2: Workshops, Demonstrations, and Tools, pp. 44-51.
9. Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E. & Peterson, A. S. (1990), Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report, SEI.
10. Kang, K.C., Kim, S., Lee, J., Kim, K., Shin, E., Huh, M. (1998). Form: A feature-oriented reuse method with domain-specific reference architectures. Annals of Software Engineering 5, pp. 143–168.
11. Lee, K., Kang, K.C., & Lee, J. (2002). Concepts and guidelines of feature modeling for product line software engineering. International Conference on Software Reuse, pp. 62–77.
12. McGregor, J.D., Muthig, D., Yoshimura, K., & Jensen, P. (2010). Guest Editors' Introduction: Successful Software Product Line Practices. IEEE Software 27(3), pp. 16-21.
13. Niu, N. and Easterbrook, S. (2008). Extracting and modeling product line functional requirements. Requirements Engineering conference (RE'08), pp. 155-164.
14. Pohl, K., Böckle, G., & van der Linden, F. (2005). Software Product-line Engineering: Foundations, Principles, and Techniques, Springer.
15. Reinhartz-Berger, I., Itzik, N., & Wand, Y. (2014). Analyzing Variability of Software Product Lines Using Semantic and Ontological Considerations. Conference on Advanced Information Systems Engineering (CAiSE'14), LNCS 8484, pp. 150-164.
16. She, S., Lotufo, R., Berger, T., Wasowski, A., & Czarnecki, K. (2011). Reverse engineering feature models. International Conference on Software Engineering, pp. 461-470.
17. Svahnberg, M., Aurum. A., & Wohlin, C. (2008). Using Students as Subjects – An Empirical Evaluation. ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 288-290.
18. Weston, N., Chitchyan, R., & Rashid, A. (2009). A framework for constructing semantically composable feature models from natural language requirements. Software Product Line Conference (SPLC'09), pp. 211-220.

Model-Driven Cross-Platform Apps: Towards Business Practicability

Tim A. Majchrzak[1], Jan Ernsting[2], and Herbert Kuchen[2]

[1] ERCIS, University of Agder, Kristiansand, Norway
[2] ERCIS, University of Münster, Münster, Germany
{tima,jan.ernsting,kuchen}@ercis.de

Abstract. Due to the incompatibility of mobile device platforms such as Android and iOS, apps have to be developed separately for each target platform. Cross-platform development approaches based on Web technology have significantly improved over the last years. However, since they do not provide native apps, these frameworks are not feasible for all kinds of business apps. Moreover, the way apps are developed is cumbersome. Advanced cross-platform approaches such as MD^2, which is based on model-driven development (MDSD) techniques, are a much more powerful yet less mature choice. We introduce MD^2 as one solution to fulfill typical requirements of business apps. Moreover, we highlight a business-oriented enhancement that further increases its business practicability.

Keywords: Cross-platform, MDSD, app, business app, mobile

1 Introduction

Businesses increasingly embrace mobile computing. Applications for mobile devices (*apps*) such as smartphones and tablets are not only developed for sale or to directly earn money with them (e.g. by placing advertisements in them). Rather, enterprises have identified usage scenarios in internal utilization by employees, field service, sales, and customer relationship management (CRM) [20]. Besides some other topics such as security [9] and testing [24], cross-platform development is a major concern [14].

The need for cross-platform development approaches arises from the incompatibility of today's platforms for mobile devices. With Apple's iOS, Google's Android, Microsoft's Windows Phone, and RIM's Blackberry (cf. e.g. [12]) there are at least four major platforms that need to be supported in order to reach *most* potential users of an app. Each platform has an ecosystem of its own and differs with regard to programming language, libraries, and usage of device-specific hardware – to name just a few factors. Developing separately for each platform currently is the choice; it is an error-prone and extremely inefficient procedure which becomes particularly frustrating when updating existing apps.

Based on the proliferation of adequate frameworks [17], mobile Webapps have become very popular. Cross-platform approaches based on Web technology such as Apache Cordova [1] (a.k.a. PhoneGap [22]) are suitable for many app projects. They are rather easy to learn, offer good community support and rich literature, and – most notably – can be deployed as *real* apps. This also allows offering them in app stores and to virtually use them in any way native apps could be used. However, apps based on Web technology are not feasible in all cases [6].

When working with Web apps, their origin in Web technology cannot be fully neglected. This has been called an *uncanny valley* [10]: a typical Webapps' look & feel is almost real but the app is slightly less responsive. Moreover, even with HTML5 [18] not all device-specific features are supported. Connecting to server-backends, as required for most *business apps* [16], is cumbersome and typically inefficient. Finally, apps are developed with a very low level of abstraction. Domain-specific knowledge has to be communicated to developers instead of being built directly into an app; existing models e.g. of business processes or information systems cannot be used even if they would be applicable to the scenario that an app is intended for.

To close the above sketched gap, we have compiled requirements for typical apps used by businesses for purposes different to sales of these apps. To enable effective cross-platform development of business apps, our group has developed a prototype for model-driven development (MDSD [25] of apps).

This paper is structured as follows. Section 2 discusses different approaches that can be used for cross-platform development. MD^2, our approach to develop cross-platform apps, is introduced in Section 3. Section 4 characterizes the particularities of a business-oriented enhancement for MD^2. In Section 5, we draw a conclusion and sketch the path for MD^2.

2 Existing Approaches

Cross-platform app development approaches have been discussed as early as 2009. Miravet et al. present their framework DIMAG, which is based on State Chart eXtensible Markup Language (SCXML) [21]. Even what is considered a cross-platform approach might vary – practitioners sometimes employ loser definitions (cf. the comparison by [7]). Apache Cordova [1] utilizes Web technology but also supports accessing native device features. Yet, its Web foundations negatively impact aspects such as app responsiveness. Other approaches build upon a *self-contained runtime* operating on custom scripting languages such as Titanium [2].

Cabana [8], AXIOM [19], and applause [3] are *generative approaches.* Cabana focuses on app development in the context of higher education. It utilizes a GUI to manipulate graphic models of app representations and allows to implement customized code. However, interactions with backends are neglected and using other platforms for achieving this is advised (cf. [8, p. 533f.]). Cabana apparently has been discontinued [26]. AXIOM takes a technical stance as it features aspects of UML and uses the programming language Groovy [13]. Moreover, it does not fully automate intermediate steps of code generation [15]. applause is most similar to MD^2: it provides a DSL, too. Yet, it is mostly restricted to displaying information and does not provide a DSL tailored to describing business apps.

3 Cross-Platform Development with MD^2

3.1 Introductory Example

A corporation has a customer relationship management (CRM) system that it wants to provide access to its sales representatives so that they can record prospective customers as part of their acquisition process through their mobile devices. However, the CRM does not support mobile devices but offers an application programming interface (API) for third party applications. In addition, the corporation has no prior knowledge of mobile app development and wants to use MD^2 to integrate its CRM and the mobile apps that are to be created. The scope of the example is limited to keep it brief: it will exclusively focus on recording prospective customers through mobile apps.

We now successively introduce MD^2's architecture, features, and domain specific language.

3.2 Architecture and Features of MD^2

Using the textual MD^2-DSL the corporation defines a model. From that MD^2 model the artifacts in the shaded area of Figure 1 are generated. In fact, these generated artifacts represent executable code for the mobile apps as well as the backend and expose the following properties:

– Mobile apps are automatically linked to the backend.
– Generated MD^2 backend already constitutes a fully Java Enterprise Edition (JEE) compliant application container including an entity model that can be persisted through Java's Persistence API (JPA).
– Developers only implement the "glue code" in the generated MD^2 backend to link it to the corporation's CRM API (corresponds to the link from MD^2 to the CRM in Figure 1). This typically is a straightforward task.

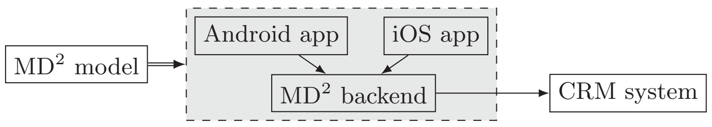


Fig. 1. Basic Architecture

3.3 MD^2-DSL

A MD^2-DSL model is structured according to the well-known *model-view-controller* pattern [5]. Thus, it consists of three parts specifying the model, view, and controller component of an app. In our example, the model (see Listing 1.1) defines just a single entity type `Contact` (with first name, surname, etc.) and an enumeration type `AcquisitionState`.

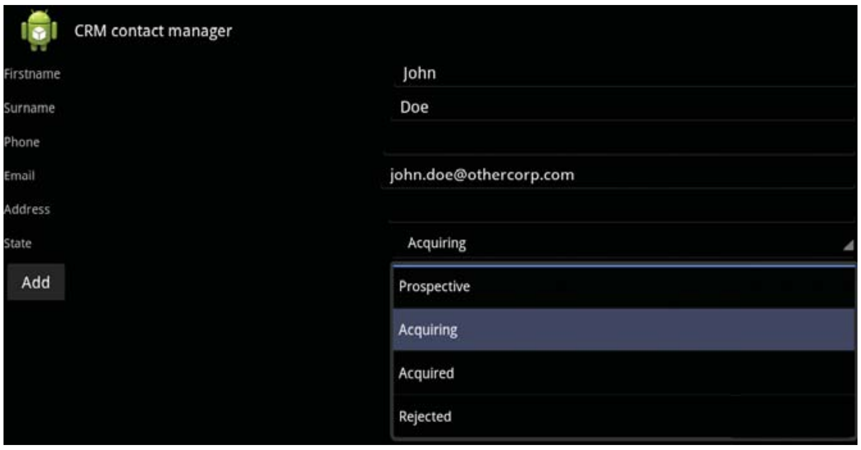
```
entity Contact {
  firstname : string
  surname : string
  phone : integer (optional)
  email : string (optional)
  state : AcquisitionState
}

enum AcquisitionState {
  "Prospective", "Acquiring", "Acquired", "Rejected"
}
```

Listing 1.1. MD^2 model

As depicted in Listing 1.2, the corresponding view first fixes a layout. Here, `FlowLayout` has been chosen, which displays the different widgets from top to bottom and from left to right on the screen. The view component `addContactView` displays a contact and a button `Add`. Here, the layout of a contact is automatically inferred from the structure of the results provided by a *content provider*, namely `contactContentProvider`. This content provider is defined in the controller part of the model (see Listing 1.3 – package definitions in this and all following listings have been stripped for brevity). As can be guessed from its name, this content provider will provide a contact. Thus, text fields for first name, surname, and so on as well as corresponding labels will be displayed. The semantics of the button `Add` will be determined in the controller component explained below. In Figure 2, the view that is generated from the view definition is shown.

```
FlowLayoutPane addContactView (vertical) {
  AutoGenerator autoGenerator {
    contentProvider contactContentProvider
  }
  Button addButton ("Add")
}
```

Listing 1.2. MD^2 view definition


CRM contact manager

Firstname John
Surname Doe
Phone
Email john.doe@othercorp.com
Address
State Acquiring

Fig. 2. Generated Android view

The corresponding controller component (see Listing 1.3) first specifies some meta-information (such as `appVersion` and `modelVersion`). Moreover, it determines the initial view component of the app (here: `addContactView`) and the action, which should be executed when the app is started (here: `startUpAction`). Then, it defines *content providers* and *actions*, which are executed when certain events are observed. A content provider can be e.g. located on a server. In our example, the content provider `contactContentProvider` is located on the server with URI `http://md2.crm.corp.com/`. Moreover, the initial `startUpAction` binds `addAction` to the `onTouch` event of the button `addButton` occurring in the `addContactView` described above. Thus, if this button on the screen is touched, the `addAction` will be executed. As shown in Listing 1.3, the `addAction` will cause the content of the text fields corresponding to the displayed contact to be stored by the content provider `contactContentProvider`. Thus, the inputs will be synchronized with the contents of the corresponding contact entity.

```
main {
  appName "CRM contact manager"
  appVersion "1.0"
  modelVersion "1.0"
  startView ADDCONTACTVIEW
  onInitialized startUpAction
}

contentProvider CONTACT contactContentProvider {
  providerType crmSystem
}

remoteConnection crmSystem {
  uri "http://md2.crm.corp.com/"
}

action CustomAction startUpAction {
  bind action addAction on ADDCONTACTVIEW.addButton.onTouch
}

action CustomAction addAction {
  call DataAction (save contactContentProvider)
}
```

Listing 1.3. MD^2 controller definition

3.4 Current Limitations

MD^2 is an academic prototype despite the cooperation with practitioners and its application in first practical projects. Thus, it poses some limitations. Some of these are inherent to the approach of using MDSD. Most, however, can be considered as work-in-progress boundaries that can be overcome in the future.

A detailed evaluation of MD^2 has not yet been done. Our approach has been refined and internally evaluated several times but no field study, detailed analysis, or competitive analysis has been conducted.

MD^2 is no "one-size-fits-all" approach. MDSD for apps most likely will never be suitable in some scenarios (e.g. apps that render their graphics on-the-fly). Nevertheless, you could consider MD^2 as "one-DSL-fits-most-cases".

While it might be undesirable to add custom code on the frontend (i.e. the app – the MDSD approach would become blurred), it would be helpful to have improved support for backend customizations. Possibly suitable approaches discussed in the literature are the generation gap pattern [11, pp. 571ff.], protected regions [25, p. 29], and dependency injection [23].

Testing (and *checking*) MD^2 should be significantly easier and at the same time very powerful since a model can be used as the basis of testing (cf. [4]). Nevertheless, we have not yet addressed testing explicitly.

As a final remark, there are no specific security features built into MD^2. Due to the DSL, it is hardly possible to use MD^2 maliciously anyway and business logic typically resides on the backend. With an extended evaluation by businesses, scenarios might arise that require additional security features that we did not yet consider. However, extending MD^2 in such cases should be hassle-free.

4 Business-oriented Enhancement

While the core of MD^2 has been described above, our framework offers additional features. In the following, support of *multi-valued elements* is described with special focus on their business-orientation.

When considering relationships between entities, two maximum cardinalities come to mind: single and multiple. Relationships with at most a single entity on the referenced side can already be defined through MD^2 models (e.g. one customer $\rightarrow$ one address). This did, so far, not hold true for relationships with multiple entities on the referenced side (e.g. one customer $\rightarrow$ many addresses). At first, multi-valued elements in MD^2 were implemented only to a certain degree. In fact, they were supported by the content providers but not on the view or controller level. Our recent work on MD^2 tackled this shortcoming and refined it to provide support for multi-valued elements.

Regarding our previous example, sales representatives need to get access to customer records as well as previous interactions. For that, the entity type `Contact` in Listing 1.4 is augmented with a list of interactions as denoted by the array-like syntax. In addition, the model is extended with a new entity type `Interaction` (with interaction date, occasion, etc.).

To display customer details and interactions, view and controller definitions require changes, too. Within the `contactDetailView` a `List` element is used to define a list view of customer interactions (see Listing 1.5). As defined by the `itemtext` value, the list view displays the occasion for each associated customer interaction. The controller definition is omitted here but is augmented to bind actions and data accordingly.

```
entity Contact {
  ...
  interactions : Interaction[]
}

entity Interaction {
  interactiondate : date
  occasion : string
  ...
}
```

Listing 1.4. Augmented MD^2 model

```
FlowLayoutPane contactDetailView (vertical) {
  ...
  List interactionsList {
    itemtype Interaction
    itemtext Interaction.^occasion
    listtype plain
  }
}
```

Listing 1.5. Augmented MD^2 view definition

Summing up, multi-valued elements might be omitted at first but are needed to address functional requirements typically found in business apps. We have made a suggestion how to cope with multi-valued elements in domain-specific languages for app development such as the one of MD^2.

As an interesting remark, the implementation of multi-valued elements in the generators for Android and iOS showed significant differences. While details are not within the scope of this paper, it is a good example for differences between the platforms that approaches, which provide native code, must overcome. At the same time, developers are relieved from understanding how (and why) similar concepts are treated different on distinct platforms.

5 Conclusion

MD^2-DSL was developed using a prototype based approach (from reference prototypes to a DSL). Beginning with a proof of concept, some design decisions regarding the language were not carried out in a consistent fashion. For example, we observed varying levels of abstraction regarding UI widgets. Considering the development of a DSL not as a serial but as a continuous process, these variations are to be aligned to offer more consistent DSL semantics. Thus, the DSL is a main concern of future work.

Due to the complex nature of MDSD, testing of MD^2's components (pre-processors, generators, etc.) was neglected so far. On a more user-centric level, testing of MD^2 apps could also be relevant for companies but also for a broader non-MD^2 related audience as well. Improving testability for MD^2 also allows providing stable artifacts (i.e. development tools, plugins, etc.) and thus improving accessibility of the framework for novices. Given a test suite, reproducible builds of the artifacts would further improve accessibility to MD^2. As a consequence, testing is the second we will address.

Even though native code for the two most common platforms is generated, industry partners expressed interest in generating code for other platforms, be it their own or another one (also cf. with the preceding section). To support custom generators, MD^2 needs to be modified in certain aspects. These modifications and the provision of additional generators are the third topic of future work.

Despite the merits of MDSD in app development, it is impossible to forecast whether it will become a dominating technology and the base of future app development. There is plenty of future work. Mobile computing and app development in particular will remain a challenging yet very exiting field of research.

References

1. Apache Cordova (2014), http://cordova.apache.org/
2. Appcelerator (2014), http://www.appcelerator.com/
3. applause (2014), https://github.com/applause/
4. Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press (2008)
5. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-oriented Software Architecture: A System of Patterns. Wiley, New York, NY, USA (1996)
6. Charland, A., Leroux, B.: Mobile application development: web vs. native. Commun. ACM 54, 49–53 (2011)
7. Cowart, J.: "Pros and cons of the top 5 cross-platform tools", http://www.developereconomics.com/pros-cons-top-5-cross-platform-tools/
8. Dickson, P.E.: Cabana: a cross-platform mobile development system. In: Proc. 43rd SIGCSE. pp. 529–534. ACM (2012)
9. Dye, S.M., Scarfone, K.: A standard for developing secure mobile applications. Comput. Stand. Interfaces 36(3), 524–530 (Mar 2014)
10. Fowler, M.: CrossPlatformMobile (2011), http://martinfowler.com/bliki/CrossPlatformMobile.html
11. Fowler, M.: Domain-Specific Languages. Addison-Wesley Pearson Education (2011)
12. Gartner Press Release (2012), http://www.gartner.com/it/page.jsp?id=1924314
13. Groovy (2014), http://groovy.codehaus.org/
14. Heitkötter, H., Hanschke, S., Majchrzak, T.A.: Evaluating cross-platform development approaches for mobile applications. In: LNBIP, vol. 140, pp. 120–138. Springer (2013)
15. Heitkötter, H., Majchrzak, T.A., Kuchen, H.: Cross-platform model-driven development of mobile applications with MD^2. In: Proc. SAC '13. pp. 526–533. ACM (2013)
16. Heitkötter, H., Majchrzak, T.A., Wolffgang, U., Kuchen, H.: Business Apps: Grundlagen und Status quo. No. 4 in Working Papers, Förderkreis der Angewandten Informatik an der WWU Münster e.V. (2012)
17. Heitkötter, H., Majchrzak, T.A., Ruland, B., Weber, T.: Evaluating frameworks for creating mobile Web apps. In: Proc. 9th WEBIST 2013. pp. 209–221. SciTePress (2013)
18. HTML5 (2014), http://www.w3.org/TR/html5/
19. Jia, X., Jones, C.: AXIOM: A model-driven approach to cross-platform application development. In: Proc. 7th ICSOFT (2012)
20. Majchrzak, T.A., Heitkötter, H.: Development of mobile applications in regional companies: Status quo and best practices. In: Proc. 9th WEBIST. pp. 335–346. SciTePress (2013)
21. Miravet, P., Marín, I., Ortín, F., Rionda, A.: DIMAG: A framework for automatic generation of mobile applications for multiple platforms. In: Proc. Mobility '09. pp. 23:1–23:8. ACM, New York, NY, USA (2009)
22. PhoneGap (2014), http://phonegap.com/
23. Prasanna, D.: Dependency Injection. Manning Pub (2009)
24. Schulte, M., Majchrzak, T.A.: Context-dependent testing of apps. Testing Experience pp. 66–70 (September 2012)
25. Stahl, T., Völter, M.: Model-driven software development. Wiley (2006)
26. "Twitter acquires team behind visual app creation tool cabana", http://tnw.to/e67X

CoreEAF – a Model Driven Approach to Information Systems

Tomas Jonsson[1] and Håkan Enquist[2]

[1] Genicore AB, Göteborg, Sweden
tomas@genicore.se,
WWW home page: http://www.genicore.se
[2] IT University, Göteborg, Sweden
enquist@chalmers.se
WWW home page: http://www.ituniv.se/english

Abstract. Model driven IT systems development with code generation is today used in several, mostly technical, application areas. However, for large IT based Information Systems (ITbIS) it is still quite uncommon. An innovative case of low cost and high quality ITbIS is presented along with the model driven framework applied. This innovation is made possible by a coherent model driven approach with integrated method and tool support for the complete development and maintenance cycles of an ITbIS. For over 20 years and still, the FMV ERP system has been designed, extended and modified in pace with changes in the organization as well as disruptive changes in information technology.

Keywords: Business Information System, Business Knowledge Model, Domain Model, Model Driven Design, Innovation, Information Engine

1 IT based Information Systems (ITbIS) Development

Large IT based Information System (ITbIS) projects frequently miss targets or fail completely [1] [2]. In Europe, the cost of missed targets and failures was estimated to 142 Billion Euros per year[3]. Obviously, methods in current use are not sufficient to solve the task. There are only a few examples of large ITbIS being built with a model driven approach [4] although very promising results are shown by some model driven approaches [5] [6].

Core Enterprise Architecture Framework (CoreEAF^{TM}) – a model driven ITbIS framework – comprises method, tools and platform for building and executing ITbIS from models. CoreEAF represents an interdisciplinary approach to ITbIS design, combining theories from disciplines such as management [7], information theory [8] [9], cognition [10] [11] and computer science [12].

Focus in this paper and demo is on illustrating the unbroken chain of support for model driven design and execution of ITbIS, leaving in-depth descriptions of each enabling concept and its implementation to be presented in the future.

The framework Fig. 1, is designed to produce ITbIS with a Model View Architecture, as originally proposed by Reenskaug [13]. The Business Knowledge

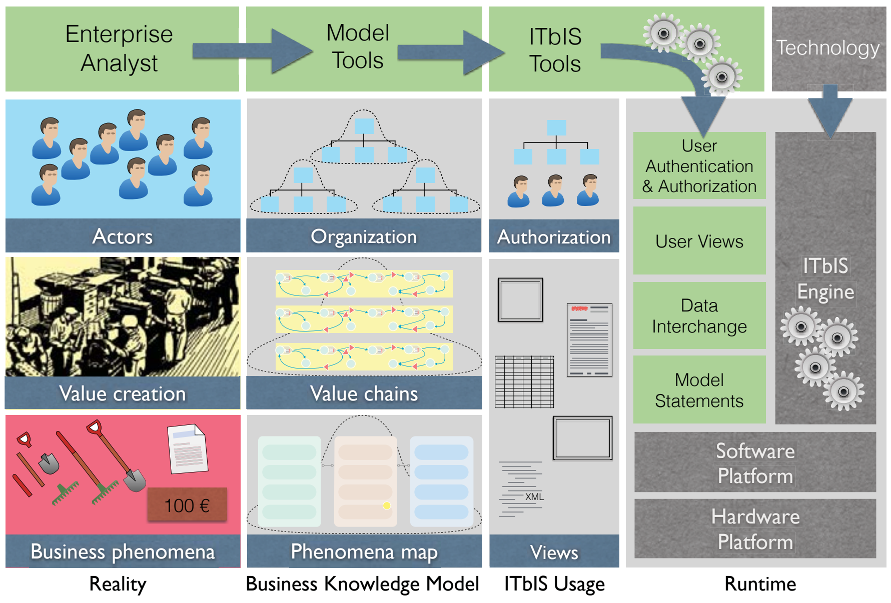


Fig. 1. Core Enterprise Architecture Framework

Model (BKM) is from a business perspective, formalized business knowledge. From a technical perspective, the BKM is an object oriented declarative model describing information structure, information processing and information rules, including information access rules. Views are defined declaratively and connected to model elements, which in runtime operates as windows for accessing information in the ITbIS. There are different types of views: graphical user interface; word processor interface; document generation; report generation and data interchange views. Each type of view is described either with a tool or a language.

Building an ITbIS with CoreEAF is done solely by declaring a BKM and it's views, excluding programming in the traditional sense.

2 ITbIS Case – ERP System for Defence Material Acquisition

The Swedish Defence Material Administration's (FMV's) Enterprise Resource Planning (ERP) system called FMV-Core, Fig. 2, is the most extensive system out of five, built and executed with the current version of CoreEAF. Applying the CoreEAF for FMV-Core is by now a proven innovation for providing ITbIS to an organization at low cost, low error rate, short lead time and high requirement fulfillment. This innovation is made possible by coherent design support for consistent model driven development and change actions throughout the ITbIS life cycle, along with a coherent runtime environment for ITBIS execution with full consistency between models and target system behavior.

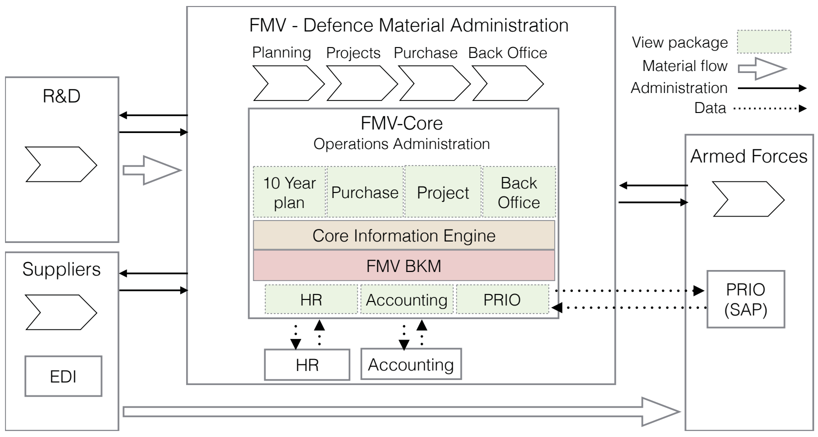


Fig. 2. Organizational structure and FMV-Core

FMV has ongoing material acquisition projects worth about 6 Billion Euros and a turnover of 2 Billion Euros per year. FMV-Core is an ITbIS with 1.200 users and handles information in FMV's core business. This includes project planning, project execution and accounting as well as complex purchasing processes from RFQ, tenders, complex contract drafting to delivery and payments. FMV-Core is integrated with several other IT systems, one of which is the Swedish Armed Forces SAP based ERP system (PRIO), using the integration facility CoreCom of the CoreEAF information engine.

Regarding design metrics, the FMV-Core system is defined with 33.000 declarative statements, whereof 13.000 statements define the BKM and 20.000 define views. FMV-Core BKM includes 230 business phenomena with 7 levels of generalization and 400 relations. The phenomena altogether carry 3000 value attributes all managed by 5600 rules and calculations.

This level of content complexity implemented in e.g. standard systems would typically include millions (>1.000.000) of lines of program code. For instance SAP is built with more than 300.000.000 lines of code [14].

3 Modeling the Enterprise – Method, Language and Tools

The key concepts applied to implement coherent support for consistent model driven design of ITbIS are described below.

3.1 Business Phenomena – Understanding the Reality of an Organization

The first and most profound aspect is the business phenomena aspect. The model of business phenomena shall directly and only directly correspond to actual business phenomena and be labelled with the business terminology.

Concrete as well as abstract business phenomena such as agreements, projects, services, etc. need to be understood and documented in the model. It is often challenging to understand and document abstract phenomena as they only exist in the minds of people and are constantly undergoing changes and redefinitions. Thus, business phenomena modelling requires fundamental understanding of human perception and of how to capture human knowledge into formal models using a modelling language.

3.2 Value Creation – What is Being Achieved

The second aspect is the value creation aspect, documented with value chains, as in Fig. 3.

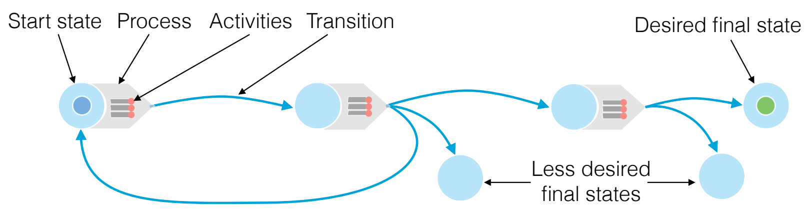


Fig. 3. Value Chain Diagram

Some of the business phenomena can be considered more or less static, i.e., they exist in the business and do not undergo any transformation of value. Other phenomena undergo value transformations, such as products being assembled in a factory. Also intangible phenomena undergo value transformations and are most likely the only kind of value transforming phenomena in organizations that produce something other than tangible products.

Examples of value transforming intangible phenomena are business agreements such as sales agreements or purchase agreements. In the public sector we can find value transforming phenomena such as claims and license applications.

For each value transforming phenomenon, a state diagram is defined representing the value chain of the phenomenon. The value chain has a start state and typically one or more final states, where one of the final states represents the desired final state of the value transformation. The intermediate states represent steps towards and sometimes away from the desired final state.

E.g. the value chain of a sales agreement can be composed of the states suggested, quoted, rejected, ordered, delivered, returned, paid and refunded, where *suggested* is the start state and *paid* is the desired final state. Note that a state transition such as *quoted* to *ordered* represents a positive value progression and *quoted* to *rejected* represent a negative value progression.

Connected to each state is a set of activities to be performed in order to reach one of the next states of the value chain. E.g. the transition *ordered* to *delivered* could be connected to several activities such as picking items in warehouse, package items, attach address label, give to post office, etc.

3.3 Organizational Structure – Roles, Responsibilities and Actors

The third aspect is the organizational structure. Business roles with responsibilities are added and connected to value chains of value transforming phenomena in the model.

Business roles can have two different kinds of responsibilities in relation to the value chains. Firstly, a role can have the overall responsibility for a certain phenomenon to progress to certain states. Secondly, for performing activities moving the phenomenon closer to a following state in the value chain. Roles can further be grouped into organizational structures representing e.g. management structure, team structure etc., depending on phenomenon, activity and organizational policies.

Finally, actors can be connected to the roles. Actors are either individuals e.g. staff members of the organization, customers, suppliers, etc. or machines performing some activity. Machines could be mechanical machines or IT systems.

3.4 Core Model Language (CML) – How to Describe BKM

Core BKMs are created and maintained in modelling tools allowing graphical editing and navigation of the model. The tools are based on a declarative language, Core Model Language (CML)

CML formalism follows the basic principles of a strongly typed class based object oriented (OO) language. However, CML is declarative and all data processing is described with parameterless functions without side effects (expressions). I.e. the value of an attribute can only be set by the attribute's value function, not by instructions in several different methods in the class. This means that each calculated value of the system is clearly defined in one place and one place only, usually as a single line of declarative statement. This gives a great advantage in terms of changeability, predictability and fault localisation. A change in one expression will only affect the values of one attribute or if a value of an attribute is incorrect, the problem will be located in its value expression.

In CML the OO *class* concept is called *phenomenon*.
Data container: A phenomenon has attributes which when instantiated can hold values.

Generalization: A phenomenon can be part of a generalization – specialization hierarchy.
Encapsulation: Stronger than in classic OO since attribute values can only be set by the attribute's value function, not by other components of the phenomenon (class).
Polymorphism: An attribute can have different definitions in a generalization – specialization hierarchy.

Additional structural components have been added to the traditional OO concept, such as relation, view and change permissions, value chain, activity, role and organizational entity.

Examples of attribute and value declarations (identifiers in ' ')
phenomenon 'person'
attribute 'full name' **value** 'given name' + " " + 'family name'
phenomenon 'order'
attribute 'order total' **value sum** 'items' [**not** 'complimentary'].'price'

The language, as shown, defines data flow graphs not execution order. It is up to the underlying information engine to resolve execution order depending on events in the data flow graphs.

3.5 Tools for Enterprise Modeling

There are two tools which can be used for modeling as described in sections 3.1-3.4. They fit in to the CoreEAF as shown in Fig. 4.
Browser: WEB based environment for creating and viewing models.
Builder: Windows based environment for creating and viewing models.

4 Modeling the ITbIS – Tools and Languages

Once the business model is created it is time to design an ITbIS that meet the information management needs of the organization and its people.

Designing the ITbIS essentially means modeling various views of the BKM. For this purpose various tools and languages are used, which assumes and is connected to the BKM. The design tools and languages architecture is shown in Fig. 4.

AppBuilder: Tool for building a component based graphical user interface, connected to a model, i.e. an application.
Reporter: User tool for creating reports with flexible search and result criteria.
IDoc-L: XML based language to define document oriented interactive user interface using MS-Word as platform.

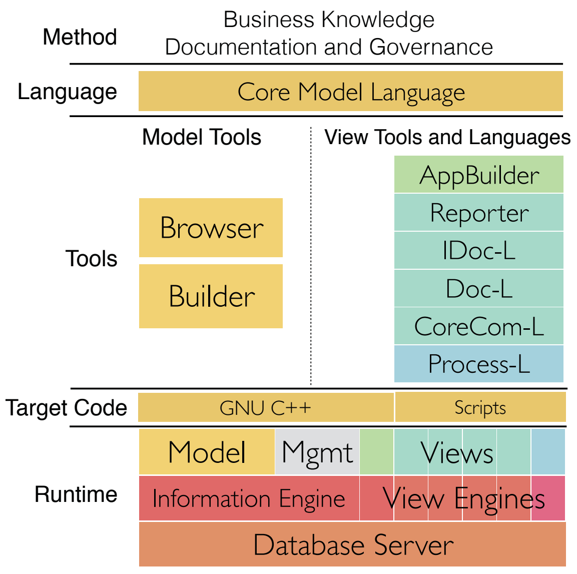


Fig. 4. Method and Tool architecture

Doc-L: XML based language to define paper documents from model data.
CoreCom-L: XML based language to define import/export of model data in XML or other text formats.
Process-L: XML based scripting language to monitor and modify data in the model based system.
Target Code The tool chain is based on C++ and executable on Linux and MS-Window. The script languages are CoreEAF specific.

5 ITbIS Runtime Components

Information Engine: The Information Engine handles persistence, information consistency, serving multiple users with real-time information while maintaining transactional integrity over a distributed network of resources, information security and a timeline for each phenomenon and its context. The information engine is programmed in C++ and uses an SQL database engine for data storage.
Applications: Model, information engine, view engines and views created by AppBuilder are compiled together into user applications. In an ITbIS there can be several different applications, where all applications have the same model but different views.
Dynamic data input and output: There are four dynamic view engines active

in runtime. One view engine which interprets report definitions and three view engines which run XML defined scripts one for each type of view tool: IDoc-L, Doc-L, CoreCom-L.
Management (Mgmt): Component for administrating users roles in the organization and for authorization in the ITbIS.

6 Current Status and Future Developments of CoreEAF

Ongoing research and publication on Model Driven ITbIS Design (MDID) include a case study and subsequent thematic studies on FMV Core.

A community for collaboration on knowledge development on MDID is being formed with academia, industry and public sector. The community will also provide knowledge resources and a web based platform for tools and experimentation.

Genicore is currently developing a new web based modeling tool with a lightweight information engine and auto generated UI for MDID to enable education and training.

References

1. L. Laird and C. Brennan : Software Measurement and Estimation A Practical Approach. IEEE Computer Society / John Wiley & Sons (2006)
2. The Standish Group: The CHAOS Manifesto. The Standish Group (2013)
3. McManus, J. & Wood-Harper T. : Understanding the sources of information systems project failure. Management Services, 51(Autumn), 3843 (2007)
4. Enquist, H. & Jonsson, T. : Sammanställning av information om Model Based System Development (MBSD) för informationssystem typ affärssystem. Research report, Genicore AB, Gothenburg (2013)
5. Pawson, R. : Naked Objects. Thesis, University of Dublin, Trinity College (2004)
6. Whittle J., Hutchinson J. Rouncefield M. : The State of Practice in Model-Driven Engineering. Software, IEEE Volume: 31, Issue: 3, Page 79 - 85 (2014)
7. Mintzberg H. : Structure in fives, designing effective organizations. Englewood Cliffs, N.J. Prentice-Hall (1983)
8. Langefors B. : Essays on Infology. Studentlitteratur: Lund (1995)
9. Enquist H. Makrygiannis N. : Understanding Misunderstandings. HICSS 82480083 (1998)
10. Hauser M.D., Chomsky N., Fitch W.T. : The faculty of language: what is it, who has it, and how did it evolve? Science vol 298 p.1569-79 (2002)
11. Kohonen T. : Associative and self organizing memory. Springer-Verlag (1988)
12. Lindskov Knudsen J., Lofgren M., Lehrmann Madsen O., Magnusson B. Et. al. : Object-Oriented Environments - The Mjølner Approach. Prentice Hall (1993)
13. Reenskaug T. : THING-MODEL-VIEW-EDITOR an Example from a planningsystem. Xerox PARC technical note. (1979) http://heim.ifi.uio.no/˜trygver/1979/mvc-1/1979-05-MVC.pdf
14. Vishal Sikka, Keynote (2008) http://scn.sap.com/docs/DOC-14735

Towards a Model of Context-Aware Recommender System

Lauma Jokste

Information Technology Institute, Riga Technical University, Kalku 1, Riga, Latvia
lauma.jokste@rtu.lv

Abstract. Users often have difficulties to use large-scale information systems efficiently because of their complexity. Additionally, these systems might be context dependent. If these context dependencies are taken into account during the system's run-time phase, the most appropriate functionality might be provided to users in the form of recommendations for each context situation. The paper proposes to account for the context dependencies by using context aware recommendations and outlines an approach for modeling such recommendations. This approach is based on methodological foundations of Capability Driven Development. The paper discusses a motivational case for developing context aware recommendations and presents the initial method for recommendation modelling.

Keywords: Recommender Systems, Recommendation Modeling, Context-Aware recommendations, Capability Metamodel.

1 Introduction and Related Work

Large-scale information systems (IS) are subject to dynamically changing circumstances in the IS delivery phase. The current situation can be described by different context data defined as *"any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between the user and the application, including the user and the applications themselves"* [1]. The context data can be taken into account in IS delivery thus increasing the usability and user satisfaction.

Complex and extensive IS might reduce the user satisfaction and, if possible, users might choose alternative ways of completing their tasks. For example, citizens are avoiding using public e-services and are favoring physical services. Recommender systems are widely used in order to improve the usage of software and tools. They provide suggestions by recommending the items that users might likely be interested in [2]. Recommender systems are increasingly popular. Many recommender systems focus on improving and evaluating the collaborative-filtering technique [3]. They use internal information about historical user activity, user profile information and other to match users with recommended items [4].

Gradually, the recommender systems start to use more varied data and data sources, for example, social network data [5][6][7]. In some cases the context information might be relevant to calculate the most appropriate recommendations for users by using context-aware recommendation systems [8]. Recent investigations present location-based [9] and weather-dependent [10] recommendation algorithms and methods. Discovering suitable context data is still a challenge in recommender systems evaluation [11]. The proposed approach in this paper includes the usage of different context data that can be retrieved from both internal and external data sources. Context processing includes not only reading the context data, but also context data analysis which helps to predict the context data and user behavior.

The recommender systems are usually item oriented and suggest the items in which users would be potentially interested in [12]. For instance, e-commerce sites such as Amazon.com recommend items users would be likely to buy [13], while content based systems recommend *things* based on textual analysis, e.g. in research area, citations can be recommended for the research by analyzing words in research papers [14]. In order to improve the recommender algorithms, hybrid recommender systems are developed by combining different recommendation algorithms and methods into one information system [15]. The approach proposed in this paper assumes that recommendations could be any kind of software entity and examples of recommendations include suggestions to execute a function, procedure, workflow or to perform data processing operations.

This paper presents the recommendation system proposal which starts with recommendation modeling. Different kind of context information can be used as an input to the recommendation module. The modeling phase is considered as significant since models can help to deal with complexity and are easy perceptible for stakeholders without any specific IT skills [16]. Recommendation modeling includes specifying a set of business rules that help to define the software entities context dependencies and appoints which recommendations should be run in each contextual situation.

Nowadays variability in IS delivery becomes more and more important. When business processes change, the software supporting these processes should be adjusted accordingly in order to fulfill the organizations goals [17]. Competitive software should be able to deal with the variability with minimal efforts. Changes in business processes can be affected by internal and/or external context. Adjusting software to changes in form of user recommendations by integrating Recommendation module in existing software allows to deal with variability without an important effort because recommendations can be easily changed in recommendation module without changing underlying software.

This paper analyses the potential of using context-aware recommendations and proposes an approach for modeling context aware recommender systems. The modeling approach is based on the Capability Driven Development (CDD) method used in development of adaptive systems [18]. The potential of using context-aware recommender systems is analyzed by exploring a use case from the e-government domain described in section 2. Section 3 provides an overview of the recommendation modeling method including a small representative example of recommendation modeling. Section 4 presents the conclusions and future work.

2 Motivational case

In order to justify development of context aware recommender systems, context dependencies in large scale IS are analyzed by using the case of the public electronic finishing service E-Loms, part of the municipal e-service platform www.epakalpojumi.lv developed by the Latvian company ZZ Dats Ltd. Currently, E-Loms provides several functions: to check information about lakes and rivers; to buy fishing licences online; to register the catch etc. E-service users' statistics (see the following figures) show that the number of licences sold varies in a wide range meaning that the usage of this service is affected by different circumstances. Several context elements affecting the e-service usage have been retrieved during the e-service usage analysis.

Assumption 1: E-service usage depends on the current and predicted air precipitation level. Fig. 1 shows the number of licences sold in September 2014 and the level of air precipitation in September 2014. There have been two pronounced rain periods and during these periods the number of fishing licences sold has rapidly decreased, thus the service usage dependents upon current and predicted air precipitation level.

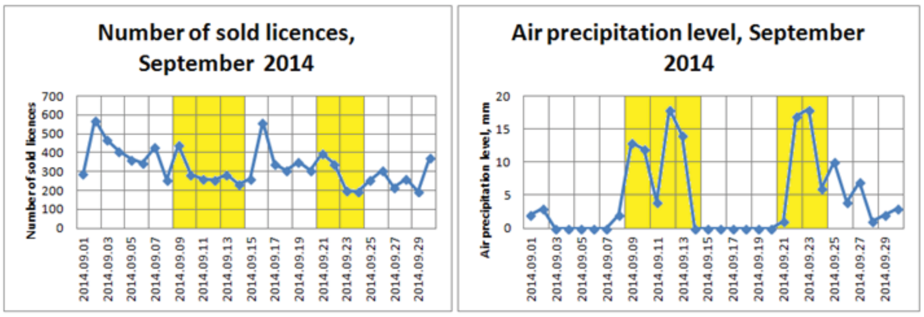


Fig. 1. E-service usage and air precipitation level dependencies

Assumption 2: E-service usage depends on the current and predicted air temperature. Fig. 2 shows the number of fishing licences sold and the average air temperature in February 2014. The same periods are highlighted in both charts. Periods with the lowest air temperature and the sharpest drops in temperature match with periods in which the largest numbers of fishing licences have been sold. In winter, ice fishing is very popular but it requires low air temperatures. This means e-service usage is highly dependent on the air temperature.

Assumption 3: E-service usage depends on calendar events. Analysis of e-service usage statistics in 2014 brings forward the following regularities:

- The beginning of January – many fishermen buy their yearly fishing licence;
- 1st of May – fishing season opening – in the end of April and the beginning of May the number of licences sold has rapidly increased;
- In the beginning of June the number of fishing licences sold has rapidly increased marking the beginning of the summer and vacations periods.
- The number of licenses sold is rapidly increasing on Fridays and Saturdays.

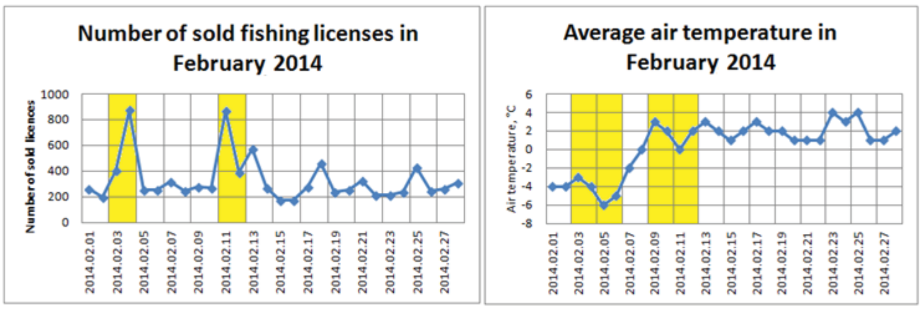


Fig. 2. Number of sold fishing licenses and average air temperatures in February 2014

The examples suggest that the usage of the E-Loms e-service is highly affected by different dynamical context data. When monitoring, predicting and taking into account the run-time context data, the e-service delivery might be improved by generating context aware recommendations for the service users.

There are many other context data that can be obtained, stored and exploited in order to apply recommendations that help users to navigate in large service catalogs or recommend users potentially the most important functions in each context situation. This could take the recommendations to a new level where context dependencies could be defined for software entities and the execution of other software entities can be provided to users in the form of recommendations.

3 Context-aware Recommendation Modeling

Usually recommendations are item-oriented where items are suggested based on user profile and historical user activity information which is mined from usage logs [19]. The use case example demonstrated in Section 2 proves that not only items can be recommended, but also different software entities can be used as recommendations.

The conceptual basis of context-aware recommendation algorithm design and development is based on the Capability metamodel [18] which is developed within the FP7 project "CaaS: Capability as a Service" and is a part of the Capability Driven Development approach. Capability term is important in recommendations modeling because context processing and recommendations application relates to IS ability to deliver a certain value to a user in changing context circumstances. This section presents the Capability metamodel which is expanded in order to fulfill the needs for modeling context-aware recommendations.

The specific aspects relevant for context-aware recommendation modeling which are added to the Capability metamodel are marked in a darker color in Fig. 3.

From the recommendation's perspective, the Software Entity is the detailed (i.e., executable) level of Capability and each Software Entity has a Context Set which defines the context information that affects the Software Entity delivery. The term Software Entity describes different executable artefacts and can be either software as a whole or a single workflow, procedure, function, job etc.

Recommendations conform to the definition of patterns which are "*reusable solutions for reaching business Goals under specific situational contexts*." [18] They are reusable solutions that can be applied to the Software Entity in a certain run-time context situation with the purpose of improving the usage of the Software Entity. The recommendations are user-oriented and improve the user satisfaction with a software, service or software entity.

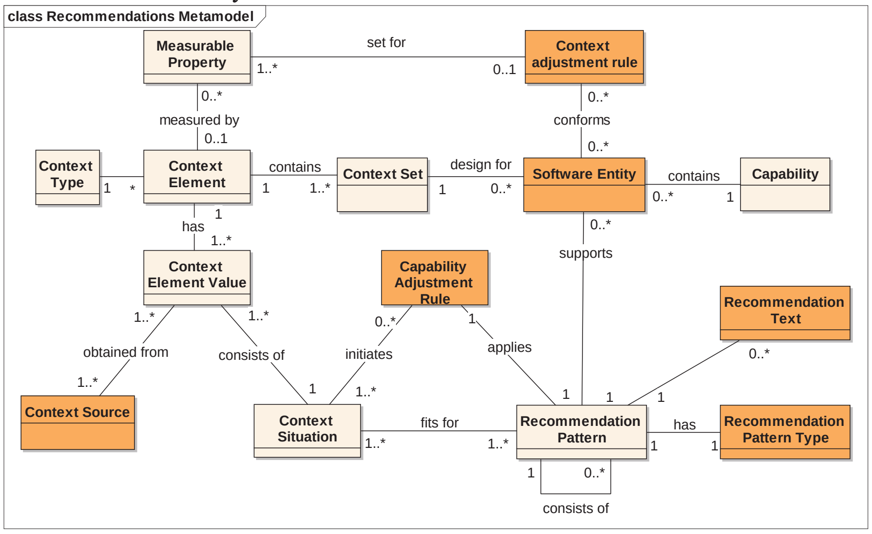


Fig. 3. Capability metamodel for modeling context-aware recommendations

A Recommendation Pattern can be applied automatically (e.g. automatic Software Entity highlighting) or recommended to a user with an informative notification which provides a choice to the user to run the recommended Recommendation Pattern. In the second case, the Recommendation Text should be defined as a separate class in the metamodel that marks separately modifiable text without changing the Recommendation Pattern itself. Each Recommendation Pattern can have multiple notification text values and the appropriate value for each context situation is defined in a Capability Adjustment Rule. In the example model (Fig. 6), the Recommendation Pattern is for instance a classificatory value 'yearly license' and the Recommendation Text recommends users to choose a licence type 'yearly license' instead of 'one day license'.

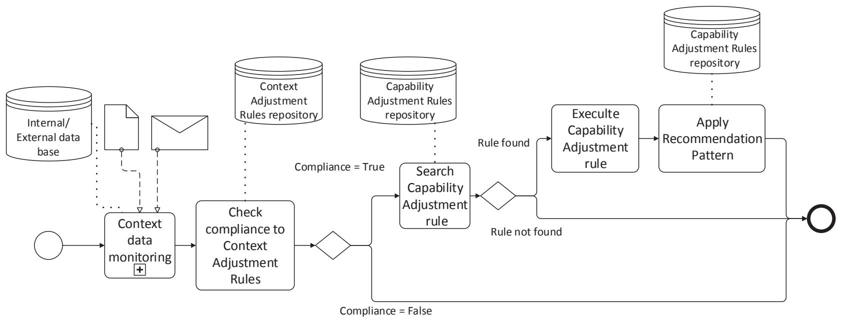

Internal/ External data base
Context Adjustment Rules repository
Capability Adjustment Rules repository
Capability Adjustment Rules repository
Context data monitoring
Check compliance to Context Adjustment Rules
Compliance = True
Search Capability Adjustment rule
Rule found
Execulte Capability Adjustment rule
Apply Recommendation Pattern
Rule not found
Compliance = False

Fig. 4. **General recommendation algorithm execution business process model**

Context Adjustment Rules define the context ranges which affect the Software Entity while Capability Adjustment Rules initiate the recommendation which is appropriate for the certain run-time Context Situation. The form of Capability Adjustment Rules should be decided, for instance rules can be written as IF/THEN or Event-Condition-Action rules. The Recommendation Pattern Type specifies the type of recommendation, e.g. a function, procedure, textual notification, etc. The Context Source serves as a reference to the source from which the context data is obtained.

Based on a Recommendation model which is modelled according to proposed metamodel, a Recommendation module can be built. In Fig. 4 a general recommendation module execution process is given. The process includes concepts from the Capability metamodel. Context data monitoring is represented as a subprocess which will be specified in further research by taking into account different context sources that can be used to obtain context information relevant to IS delivery. In Fig. 5 and 6 an example of a recommendation model is given.

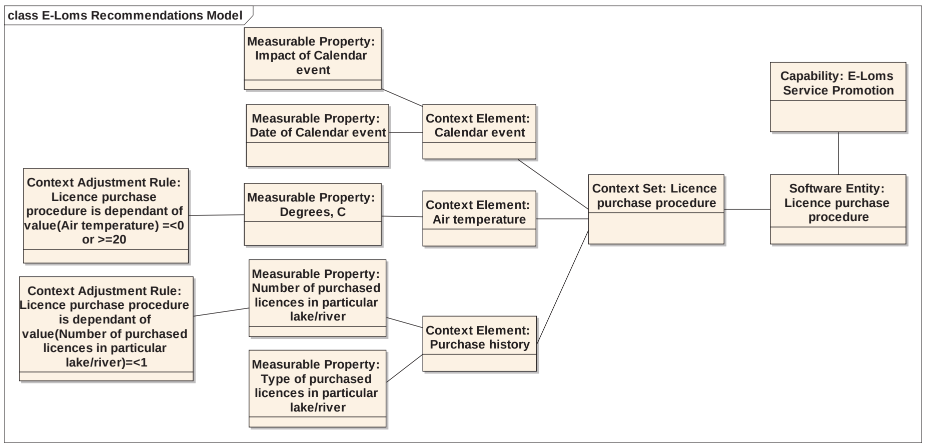

class E-Loms Recommendations Model
Measurable Property: Impact of Calendar event
Measurable Property: Date of Calendar event
Context Element: Calendar event
Capability: E-Loms Service Promotion
Context Adjustment Rule: Licence purchase procedure is dependant of value(Air temperature) =<0 or >=20
Measurable Property: Degrees, C
Context Element: Air temperature
Context Set: Licence purchase procedure
Software Entity: Licence purchase procedure
Context Adjustment Rule: Licence purchase procedure is dependant of value(Number of purchased licences in particular lake/river)=<1
Measurable Property: Number of purchased licences in particular lake/river
Context Element: Purchase history
Measurable Property: Type of purchased licences in particular lake/river

Fig. 5. Context modeling part of the E-Loms recommendation model example

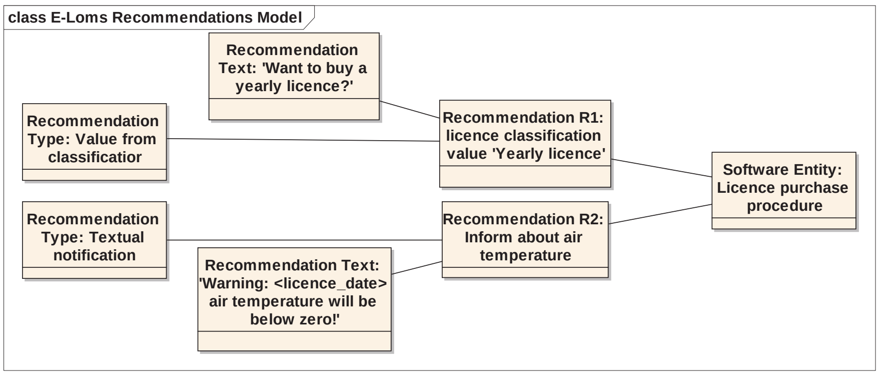

class E-Loms Recommendations Model
Recommendation Text: 'Want to buy a yearly licence?'
Recommendation Type: Value from classificatior
Recommendation R1: licence classification value 'Yearly licence'
Software Entity: Licence purchase procedure
Recommendation Type: Textual notification
Recommendation R2: Inform about air temperature
Recommendation Text: 'Warning: <licence_date> air temperature will be below zero!'

Fig. 6. Recommendation modeling part of the E-Loms recommendation model example

The Context Situation is a run-time element and cannot be modeled during the design phase. For instance, if a user is buying a one-day license in a lake where he already has bought one-day licenses, then another value from the classificatory can be recommended – ‘a yearly license’. Recommendations can be run by a Capability Adjustment Rule, e.g. *If number of bought licenses in lake X >=1, THEN run recommendation R1*.

4 Conclusion and Future work

In this paper we discuss the usage of the Capability metamodel in the design of context-aware recommendations for software entities in order to increase user satisfaction. The example demonstrated in the paper proves the potential of generating recommendations based on run-time context data.

Current recommendation algorithms and methods are mostly item-oriented and based on a limited amount of data (user profile data, user activity, similarities between users and items etc.). The delivery of many software and software entities is affected by different context information. The achievement of business goals can be improved by applying context aware recommendations in software delivery.

The recommendation modeling method would allow designing modifiable recommendations more efficiently. In the Recommendation module, recommendations can be modified by defining or changing Capability Adjustment and Context Adjustment Rules, recommendation values, or modifying recommendations in the recommendation repository. Once the module is built, stakeholders without specific IT skills should be able to manage and maintain the recommendations.

This paper presents the initial proposal for context-aware recommendations modeling. The context-aware recommendation modeling and designing method should be improved and developed in detail in following aspects:

- The form for defining the Context Adjustment Rules and the Capability Adjustment Rules should be specified;
- The context processing subprocess should be specified;
- The technical solution for developing and implementing the recommendation module should be developed;
- The recommendation modeling method and recommendation module should be validated.

References

1. Dey, A.K. Understanding and Using Context. In: Personal Ubiquitous Computing, vol 5(1), pp. 4–7 (2001)
2. Rubens, N., Kaplan, D. and Sugiyama, M. Recommender Systems Handbook: Active Learning in Recommender Systems (eds. P.B. Kantor, F. Ricci, L. Rokach, B. Shapira). Springer (2011)

3. Hang, S., Hui, P., Kulkarni, S. R. And Cuff, P.W. Wisdom of the Crowd: Incorporating Social Influence in Recommendation Models. Proc. ICPADS ’11, IEEE, pp. 835-840 (2012)
4. Burke, R. Hybrid Recommender Systems: Survey and Experiments. In: User Modeling and User-Adapter Interaction, Vol 12, Issue 4, pp 331-370 (2002)
5. Aranda J., Givoni I., Handcock, and Tarlow, D. An Online Social Network-based Recommendation System, Technical report (2010)
6. He J. and Chu, W.W. A Social Network Based Recommender System, Annals of Information Systems: Special Issue on Data Mining for Social Network Data (AIS-DMSND) (2010)
7. Shang, S., Hui, P., Kulkarni, S.R. and Cuff, P.W. Wisdom of the crowd: Incorporating Social Influence in Recommendation Models. In: Proceedings of the IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS '11), pp. 835-840 (2011)
8. Adomavicius, G. and Tuzhilin, A.Context-Aware Recommender Systems. In: Recommender Systems Handbook, pp. 217-253 (2011)
9. Savage, N.S., Baranski, M., Chavez, N.E. and Höllerer, T. I’m feeling LoCo: A Location Based Context Aware Recommendation System. Proceedings of the 8th International Symposium on Location-Based Services (2011)
10. Braunhofer, M., Elahi, M., Ge, M., Ricci, F. and Schievenin, T. STS: Design of Weather-Aware Mobile Recommender Systems in Tourism. In: Carolis B.N., Carolis E. (eds.) Artificial Intelligence meets Human Computer Interaction, CEUR Workshop proceedings, vol 1125 (2013)
11. Yujie Z. and Licai W. Some challenges for context-aware recommender systems. In: Proceedings of the 5th International Conference on Computer Science and Education (ICCSE), pp. 362 – 365 (2010)
12. Linden, G., Smith, B. and York, J. Amazon.com Recommendations. Industry report. Published by the IEE Computer Society (2003) URL: http://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf Last accessed: March 26, 2015.
13. Duma, D. and Klein, E. Citation Resolution: A method for evaluating context-based citation recommendation systems. In: Proceedings of the 52nd Annual Meeting in the Association for Computational Linguistics, Volume 2, pp. 358-363 (2014)
14. Hussein, T., Linder, T., Gaulke, W. and Ziegler, J. Hybreed: A software framework for developing context-aware hybrid recommender systems. In: User Modeling and User-Adapted Interaction, Vol 24. Issue 1-2, pp. 121-174 (2014)
15. Ricci, F., et al. (eds.). Recommender Systems Handbook, Springer Science + Business Media (2011)
16. Carvalho, J. A. Strategies to Deal with Complexity in Information Systems Development. URL: http://www3.dsi.uminho.pt/jac/SI/zdocumentos/complexity.pdf Last accessed: March 26, 2015
17. Soffer, P. Analyzing the Scope of a Change in a Business Process Model, Proceedings of Int’l Conference on Advanced Information Systems Engineering Workshops (2004)
18. Stirna, J., Grabis, J., Henkel, M. and Zdarvkovic, J. Capability Driven Development – and Approach to Support Evolving Organizations. In: The Practice of Enterprise Modeling: PoEM, Germany, Rostok, 7-8 November, 2012. Berlin: Springer Berlin Heidelberg, pp.117-131 (2012)
19. Koenigstein, N. and Koren, Y. Towars scalable and accurate item-oriented recommendations. In: Proceedings of the 7th ACM conference on Recommender systems (2013)

An Approach to Conceptual Fit Analysis for COTS Selection

Antoni Olivé

Department of Service and Information System Engineering
Universitat Politècnica de Catalunya – Barcelona Tech
antoni.olive@upc.edu

Abstract. One aspect that has received little attention in COTS selection is what we call conceptual fit, which we evaluate in terms of the existing misfits. We define the notion of conceptual misfit and we present an approach that determines the conceptual misfits between the user requirements and a set of candidate systems. The approach consists in defining a superschema, the mapping of the conceptual schemas of the candidate systems and of the user requirements to that superschema, and the automatic computation of the existing conceptual misfits. We believe that conceptual fit could be taken into account by almost all existing COTS selection methods.

Keywords. Conceptual Fit, COTS selection, Conceptual Modeling

1. Introduction

Nowadays, many organizations build many of their information systems by customizing and/or integrating Commercial-off-the-shelf (COTS) systems. Selecting the most convenient COTS system for a particular situation has become a critical activity in information systems engineering.

COTS system selection essentially consists in evaluating user requirements with respect to characteristics of candidate systems. The evaluation is performed by defining a set of criteria, assessing the importance of each criterion for the users and the degree to which the criterion is satisfied by a system. One kind of criterion that has received little attention is what we call conceptual fit. It is similar to what is called domain compatibility in OTSO [1] and suitability of data in GOThIC [2].

This paper analyzes the conceptual fit between user requirements and COTS systems. We define the notion of *conceptual misfit* and we present an approach to the determination of the existing conceptual misfits between a set of user requirements and a system. The absence of conceptual misfits indicates a perfect conceptual fit. Our notion of conceptual misfit has been inspired in the ontological expressiveness analysis [3].

The structure of the paper is as follows. Next section identifies the different kind of conceptual misfits that may exist between a set of user requirements and a COTS system. Section 3 formalizes the general problem of evaluating the conceptual fit. In section 4 we describe the approach we propose for solving that problem. Finally, section 5 summarizes the conclusions.

2. Conceptual Fit

By conceptual fit we mean the fit between two structural conceptual schemas. In our context, one conceptual schema is that of the user requirements and the other one is that of a particular COTS system. For the purposes of this paper, we will assume simple structural conceptual schemas consisting only of entity types, ISA hierarchies, attributes and binary associations. This can be easily extended, if desired [4].

In the UML metamodel *M* (see Fig. 1) of the schemas that we consider in this paper, entity types have a name, may be abstract or concrete, may be a singleton or be unconstrained, and may have sub/supertype associations between them. Entity types may have attributes, which are properties. Properties have a minimum and a maximum cardinality, and a type. Cardinalities may be zero, one or unconstrained. Associations have two ordered participants, each of which is a property, as before.

Assume now that we have two instances of *M* that we call U_i (for user requirements) and S_j (for COTS system). We are interested in knowing how well U_i and S_j fit each other. To this end, we try to see whether there are misfits between them. Based on the simple metamodel *M* we identify three kinds of misfits in the schema elements, called deficits, incompatibilities and excesses, which we define in the following. The idea is that the degree of fit of U_i and S_j is inversely proportional to the number of misfits, the maximum being the absence of them.

2.1 Entity Type Misfits

We say that there is an *entity type deficit* between U_i and S_j with respect to (wrt) *E* if *E* is a concrete entity type of U_i but *E* is not an entity type of S_j. Note that we consider only the concrete entity types of U_i because these are the ones of interest to the users. Abstract entity types in U_i are unions of concrete ones.

There is an *entity type cardinality incompatibility* between U_i and S_j wrt *E* if *E* is a concrete entity type of U_i and an entity type of S_j, but *E* is unconstrained (not a singleton) in U_i and a singleton in S_j. Both U_i and S_j have the entity type *E* but, in U_i, *E* may have several instances while only one instance is allowed in S_j.

We say that there is an *entity type excess* between U_i and S_j wrt *E* if *E* is a concrete entity type of S_j but *E* is not an entity type of U_i. In this case, S_j includes an entity type that is not of interest to U_i.

2.2 Attribute Misfits

There is an *induced attribute deficit* between U_i and S_j wrt *A* if *A* is an attribute of the concrete entity type *E* in U_i and there is an entity type deficit between U_i and S_j wrt *E*. In this case, the deficit is induced by the entity type deficit.

There is an *attribute deficit* between U_i and S_j wrt A if A is an attribute of the concrete entity type E in U_i, S_j includes E, but S_j does not include A.

There is an *attribute cardinality incompatibility* between U_i and S_j wrt A if A is an attribute of the concrete entity type E in U_i, S_j includes A, but the cardinalities are incompatible. An incompatibility arises when the minimum cardinality in U_i is zero and one in S_j, or when the maximum cardinality is unconstrained in U_i and one in S_j.

There is an *induced attribute excess* between U_i and S_j wrt A if A is an attribute of the concrete entity type E in S_j and there is an entity type excess between U_i and S_j wrt E. In this case, the excess is induced by the entity type excess.

There is an *attribute excess* between U_i and S_j wrt A if A is an attribute of the concrete entity type E in S_j, U_i includes E, but U_i does not include A. In this case, S_j includes an attribute that is not of interest to U_i.

2.3 Association Misfits

There is an *induced association deficit* between U_i and S_j wrt R if R is an association between the concrete entity types E_1 and E_2 in U_i, and there is an entity type deficit between U_i and S_j wrt E_1 or E_2. In this case, the deficit is induced by the entity type deficits.

There is an *association deficit* between U_i and S_j wrt R if R is an association between the concrete entity types E_1 and E_2 in U_i, S_j includes E_1 and E_2, but S_j does not include R.

There is an *association cardinality incompatibility* between U_i and S_j wrt R if R is an association between the concrete entity types E_1 and E_2 in U_i, S_j includes E_1 and E_2, but the cardinalities of one of its participants are incompatible. An incompatibility arises when the minimum cardinality in U_i is zero and one in S_j, or when the maximum cardinality is unconstrained in U_i and one in S_j.

There is an *induced association excess* between U_i and S_j wrt R if R is an association between the concrete entity types E_1 and E_2 in S_j, and there is an entity type excess between U_i and S_j wrt E_1 or E_2. In this case, the excess is induced by the entity type excess.

There is an *association excess* between U_i and S_j wrt R if R is an association of the concrete entity types E_1 and E_2 in S_j, U_i includes E_1 and E_2, but U_i does not require R.

3. Evaluating the Conceptual Fit Criterion for COTS Selection

The general problem of evaluating the conceptual fit criterion can be defined as follows:

Given:

- The user requirements U_i of a system in some domain and
- A set $S_1,\ldots,S_n$ of n candidate COTS systems in that domain,

Determine:

- The conceptual misfits (deficits, misfits and excesses as defined in the previous section) between U_i and each of the $S_1,\ldots,S_n$.

Conceptual fit analysis can be performed considering the complete set of user requirements U_i and of the candidate systems $S_1,\ldots,S_n$, or considering only a fragment of them. The latter possibility is likely to be of much more practical interest in most cases.

The set of conceptual misfits found can be used as a basis for selection. If there are no misfits between U_i and S_j, then there is a perfect fit between them.

If there are one or more deficits or incompatibilities between U_i and S_j, then the selection of S_j would require either the change of the user requirements U_i (changing their intended way-of-working) or a customization of S_j for the user (customizing existing systems to accommodate users' requirements).

If there are one or more excesses between U_i and S_j, then the selection of S_j would imply dealing with the unneeded features related to those excesses, and the need of the corresponding resources.

4. A Method for Determining the Conceptual Fit

A straightforward approach to the solution of the general problem of determining the conceptual fit would be to consider each S_j ($j = 1,\ldots,n$) separately, and determine the conceptual misfits between U_i and S_j as indicated in Sect. 2. When the number n is large and/or the conceptual schemas are large, the evaluation effort may be large too.

However, in a context where the selection process must be performed several times with the same set of candidate systems $S_1,\ldots,S_n$, with different user requirements U_i, then a better solution would be to build an intermediate superschema S. That superschema S should integrate $S_1,\ldots,S_n$ in a way such that U_i and each of the $S_1,\ldots,S_n$ could be mapped to S, and such that the conceptual misfits of U_i and each of the $S_1,\ldots,S_n$ could then be computed automatically.

Based on the above idea, the method we propose consists of four parts:

1. A superschema S that is a union of all schemas $S_1,\ldots,S_n$ and all possible user requirements $U_1,\ldots,U_m$ in a given domain.
2. The definition of the schemas $S_1,\ldots,S_n$ in terms of S.
3. The definition of user requirements U_i in terms of S.
4. The (automatic) computation of the misfits between U_i and $S_1,\ldots,S_n$.

We describe these parts in the following.

4.1 The superschema

In our method, the superschema S is an instance of the metamodel M for a domain D such that:

- S includes the schemas of all possible COTS systems $S_1,\ldots,S_n$ in D.
- S includes all possible conceptual user requirements $U_1,\ldots,U_m$ in D.

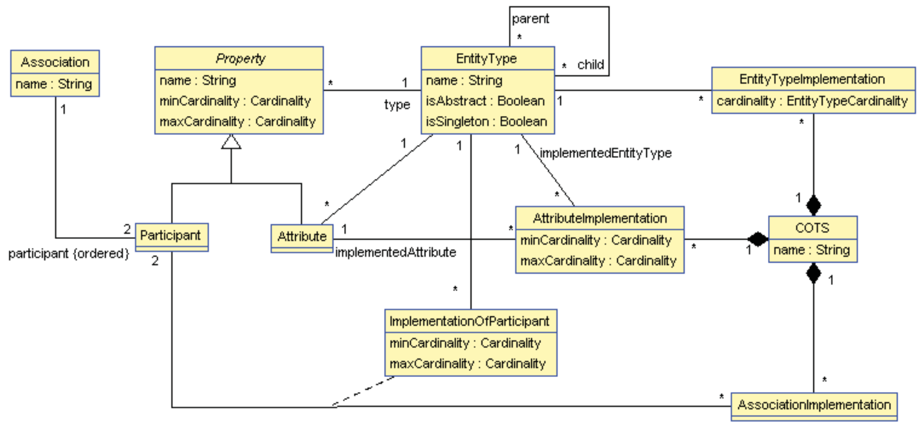


Fig.1. COTS implementation of a superschema

By inclusion of schemas here we mean that:

- *S* comprises all concrete entity types, attributes and associations that may be required by $U_1,\ldots,U_m$. On the other hand, the cardinalities of the attributes and associations in *S* must not be incompatible with those that may be required by $U_1,\ldots,U_m$.
- *S* comprises all concrete entity types, attributes and associations that are implemented in $S_1,\ldots,S_n$. On the other hand, the cardinalities of the attributes and associations in *S* must not be incompatible with those that are implemented in $S_1,\ldots,S_n$.

4.2 Mapping Conceptual Schemas of COTS Systems to the Superschema

For the purposes of conceptual fit analysis we need to know for each S_j ($j = 1,\ldots,n$) in *D*:

- The entity types of *S* implemented in S_j and their corresponding cardinalities. We are interested only in the entity types that are concrete in S_j. If S_j implements all subtypes of an abstract entity type *E* in *S*, then S_j also implements *E*.
- The attributes and associations of *S* implemented in S_j and their corresponding cardinalities.

Figure 1 shows the metamodel *M* and the extension needed to represent the part of *S* that is implemented by S_j. A COTS system is assumed to implement a set of concrete entity types (with a cardinality that may be Singleton or Unconstrained), a set of attributes and a set of associations.

Note that if *S* includes an abstract entity type *E* with subtypes $E_1,\ldots, E_m$ and *E* has an attribute *A*, then a system S_j that implements two or more of those subtypes could implement *A* differently in each case. Our metamodel of Fig. 1 takes this possibility into consideration by indicating in *AttributeImplementation* the implemented entity type. A similar reasoning applies to the association participants.

The mapping process can be superschema-driven or system-driven. In the former, the elements of *S* are taken in some convenient order, and for each of them it is

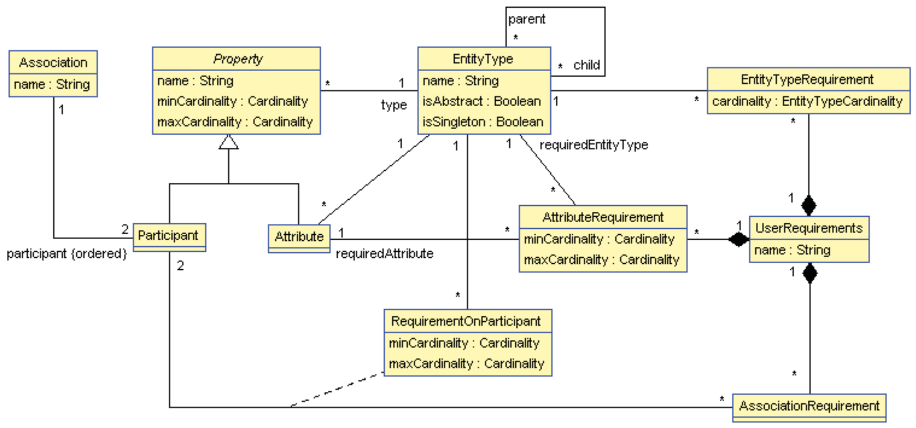


Fig.2. Extension of the metamodel *M* with user requirements

checked whether or not it is implemented by the system. If the element is a concrete entity type that is not implemented by S_j then there is no need to check the implementation of its attributes and associations. To use this process, the conceptual schema of S_j needs not to be explicit; what is needed to know is what entity types, attributes and associations of *S* are implemented in S_j.

In the system-driven process, the elements of the conceptual schema of S_j are taken in some convenient order, and each of them is mapped to *S*. To use this process the conceptual schema of S_j must be explicit.

4.3 Defining Conceptual User Requirements

For the purposes of conceptual fit analysis of U_i we need to know:

- The entity types of *S* required by U_i and their corresponding cardinalities. We need to know only the entity types that are concrete in U_i. If U_i requires all subtypes of an abstract entity type *E* in *S*, then U_i also requires *E*.
- The attributes and associations of *S* required by U_i and their corresponding cardinalities.

Figure 2 shows the extension of the metamodel *M* needed to represent the user requirements in terms of *S*. User requirements are assumed to consist of concrete entity types (with a cardinality that may be Singleton or Unconstrained), a set of attributes and a set of associations.

Note that similarly to the previous case, if *S* includes an abstract entity type *E* with subtypes E_1,…, E_m and *E* has an attribute *A*, then if U_i requires two or more of those subtypes, it could require *A* differently in each case. The same applies to association participants.

As in the mapping of systems, the definition of user requirements can be superschema-driven or requirements-driven.

4.4 Computing Misfits

In our method, once we have defined the instance of M corresponding to the superschema S for a domain D, the instances of the candidate COTS systems $S_1,\ldots,S_n$ in D and their mapping to S (Fig. 1), and the instance of the user requirements U_i and its mapping to S (Fig. 2) we can then automatically compute the misfits between U_i and $S_1,\ldots,S_n$. In what follows we explain the details of the computation in terms of the UML schemas shown in Figs. 1 and 2.

Entity type deficit. Let E be an entity type required by U_i. There is a deficit of E in S_j if E is not implemented in S_j. E can be implemented in S_j directly or by exclusion. There is a direct implementation when E is also an entity type of S_j. There is an implementation by exclusion when there is an entity type E' implemented by S_j such that E' is a supertype of E, $E_1,\ldots, E_p$ ($p > 0$) and $E_1,\ldots, E_p$ are not required by U_i. The exclusion of $E_1,\ldots, E_p$ by U_i implies that the population of E and E' will always be the same, and therefore E' can implement E in S_j.

Entity type incompatibility. Let E be an unconstrained entity type required by U_i. There is an incompatibility when E is implemented by a singleton entity type in S_j.

Entity type excess. Let E be an entity type in S_j. There is a misfit of this kind when E does not implement any entity type in U_i.

Induced attribute deficit. This happens when U_i requires an attribute of entity type E and there is an entity type deficit between U_i and S_j wrt E.

Attribute deficit. This happens when U_i requires an attribute A of an entity type E that is implemented in S_j, but that implementation does not include A.

Attribute cardinality incompatibility. This happens when the cardinalities of an attribute required by U_i are incompatible with those of its implementation in S_j.

Induced attribute excess. Let A be an attribute of a concrete entity type E in S_j. There is a misfit of this kind when E is an entity type excess for U_i. In OCL:

Attribute excess. Let A be an attribute of a concrete entity type E in S_j. There is a misfit of this kind when E is an implementation of an entity type required by U_i but A is not implemented.

Induced association deficit. There is misfit of this kind when U_i requires an association R between the concrete entity types E_1 and E_2 and there is an entity type deficit between U_i and S_j wrt E_1 or E_2.

Association deficit. There is misfit of this kind when U_i requires an association R between the concrete entity types E_1 and E_2 that are implemented in S_j, but S_j does not include R.

Association cardinality incompatibility. This happens when the cardinalities of an association required by U_i are incompatible with those of the implemented association in S_j.

Induced association excess. Let R be an association between the concrete entity types E_1 and E_2 in S_j. There is a misfit of this kind when E_1 and E_2 are an entity type excess for U_i.

Association excess. Let R be an association between the concrete entity types E_1 and E_2 in S_j. There is a misfit of this kind when E_1 and E_2 are implementations of entity types in U_i but R is not.

5. Conclusions

We have proposed a new aspect for COTS system selection, which we call conceptual fit, which assesses the fit between the conceptual structure of a given system and of the user requirements. We have identified three kinds of misfits in the schema elements, called deficits, incompatibilities and excesses. The idea is that the degree of conceptual fit is inversely proportional to the number of misfits, the maximum being the absence of them.

We have formally defined the general problem of evaluating the conceptual fit between the user requirements and a set of COTS systems, and we have proposed a new method for its solution. The method consists in defining a superschema, the mapping of the conceptual schemas of the candidate systems and of the user requirements to that superschema, and the computation of the conceptual misfits.

The main effort required by our method is the development of the superschema and the mapping of the candidate systems to it. However, this must be done only once per domain and the result could be reused in all COTS selections of a domain.

Acknowledgments. This work has been partly supported by the Ministerio de Economía y Competitividad and FEDER under project TIN2008-00444, Grupo Consolidado.

References

1. Kontio, J.: OTSO: A Systematic Process for Reusable Software Component Selection. University of Maryland Technical Reports. College Park, University of Maryland. CS-TR-3478, UMIACS-TR-95-63, (1995)
2. Ayala, C.P., Franch, X.: Domain Analysis for Supporting Commercial Off-the-Shelf Components Selection. In: D.W. Embley, A. Olivé, and S. Ram (Eds.): ER 2006, LNCS 4215, pp. 354 – 370, (2006)
3. Wand, Y.: Ontology as a foundation for meta-modelling and method engineering. Information & Software Technology 38(4): 281-287 (1996)
4. Olive, A.: Conceptual Modeling of Information Systems. Springer, Berlin (2007)

A Resource Oriented Architecture to Handle Data Volume Diversity

Pierre De Vettor[1], Michaël Mrissa[1], and Djamal Benslimane[1]

Université de Lyon, CNRS
LIRIS, UMR5205, F-69622, France
Lyon, France
`firstname.surname@liris.cnrs.fr`

Abstract. Providing quality-aware techniques for reusing data available on the Web is a major concern for today's organizations. High quality data that offers higher added-value to the stakeholders is called smart data. Smart data can be obtained by combining data coming from diverse data sources on the Web such as Web APIs, SPARQL endpoints, Web pages and so on. Generating smart data involves complex data processing tasks, typically realized manually or in a static way in current organizations, with the help of statically configured workflows. In addition, despite the recent advances in this field, transfering large amounts of data to be processed still remains a tedious task due to unreliable transfer conditions or transfer rate/latency problems. In this paper, we propose an adaptive architecture to generate smart data, and focus on a solution to handle volume diversity during data processing. Our approach aims at maintaining good response time performance upon user request. It relies on the use of RESTful resources and remote code execution over temporary data storage where business data is cached. Each resource involved in data processing accesses the storage to process data on-site.

Keywords: resource oriented architecture, data integration, data semantics, smart data

1 Introduction

During the last few years, governments, companies and organizations have opened their databases and information systems to the world across the Web, thanks to initiatives such as the open data project [8]. These data sources are typically exposed via Web APIs [11] or SPARQL endpoints and can be combined in service mashups [1] to produce highly valuable services. As an/For example, the sets of APIs provided by Twitter, Amazon, Youtube or Flickr are reused/used again in thousands of mashups[1].

This smart use of data has caught the interest of the community as a natural development. The objective of smart data [13] is focused on producing high-quality data that is directly useful to users. Automatically integrate data from

[1] See also http://www.programmableweb.com/

diverse sources in order to produce smart data is currently a hot research topic. Despite these advances, data quantity still hampers data exchanges and remains a bottleneck in architectures, especially when it comes to data transfer between resources. By taking advantage of REST architectural style and the use of resources, we unfortunately suffer from the limitations of Web protocols, and it is sometimes difficult to transfer large quantities of data through HTTP. There is a need for an approach that minimizes data transfer and performs data processing tasks closer to the data source to reduce network traffic.

In this paper, we present our adaptive architecture and propose a solution, through the use of adaptive data access strategies and remote code execution on temporary data storage unit resources, to handle latency and architecture issues that appear when processing data volume diversity. Our architecture optimizes and adapts workflows to handle the variety of data sources involved in the query. In this approach, we present this resource oriented architecture and explain our solution for reducing latency in resource oriented architecture.

This paper is organized as follows. Section 2 presents related approaches to handle volume diversity in data sources in RESTful architectures. Section 3 presents our resource oriented architecture and our solution to minimize data exchange. Section 4 gives an evaluation of our prototype in terms of responsiveness and shows how it responds to user requests with acceptable timings. Section 5 discusses our results and provides guidelines for future work.

2 Related Work

Over the past few years, big data has generated a lot of interest from researchers and industrials, answering to four challenges, known as the four Vs: Volume, Variety, Velocity and Veracity. Defining our smart data architecture [3], we provide data source models and strategies to support the Variety challenge and analysis, combination and data cleaning for Veracity. Finally, Velocity is managed by adapting workflows and optimizing resource orchestration at runtime to improve response time. On top of that, we put more focus on data quality through the use of semantics, metadata and intelligent processing techniques. In this smart data context [13], propositions focus on data quality rather than quantity and build smarter architecture. In this data-driven context, Web standards comes as a solution, but limits data exchanges. The following appproaches have addressed the data issue in HTTP-based solutions.

Devresse et Al. [4] propose a library called *libdavix* allowing high performance computing world to benefit from HTTP and the RESTful principles. This approach focus on adapting the HTTP protocol, maximizing the reutilization of TCP connections, by providing a dynamic connection pool coupled with the usage of the HTTP Keep-Alive feature. By avoiding useless protocol handshakes, reconnections and redirections, their approach improves efficiency of large data transport through HTTP. In *Fast Optimised REST (FOREST)* [9], Ko et Al. propose a non-invasive technique relying on TCP data encapsulation in UDP-based data transfer payloads [6]. Evaluations shows good results, but the approach does not seem to provide a real solution, it is a low level fixing to benefit from advantages of other protocols. Zheng et al. [15] provide an overview of service-generated big data as well as big data-as-a-service, a flexible infrastructure providing common big data management functionalities. Their approach rely on cloud computing techniques to handle collection, storing, processing and visualization of data and they address some significant challenges, particularly about variety or volume and how infrastructure must support (and adapts) this variety and volume to provide fast data access and processing. Van Der Pol et Al. [14] propose an approach based on multiple paths TCP [5] to transfer huge data sets over networks. Their approach relies on load balancing transfer through the different available paths relying on parallel multiple TCP requests. Their approach can handle different paths with different bandwidths, balanced over the different interface offered by the system. They propose a prototype According to these approaches, it becomes clear that the most powerful solution is to minimize data transfers, process data volumes closer from the source to reduce the unnecessary traffic. In the next section, we present our resource-oriented architecture, the models that helps to build it, and finally, we present our solution to handle data volume diversity in our smart data architecture.

3 Contribution

In order to handle our smart data challenges, we envision a resource-oriented architecture, generating adaptive workflows at runtime to adapt to data source characteristics. We rely on the data source models presented in our previous work [3] to represent data source characteristics such as uri, request format, volume, latency, etc. Relying on these models, we are thus able to generate adaptive *data processing workflows* according to characteristics that appears on data source descriptions.

Then we define an resource oriented architecture to manage these adaptive workflows and the resources involved.

3.1 Architecture

In our approach, we define different necessary steps to complete the data aggregation process and produce smart data. The main steps are **extraction** from data source and transformation into a pivot format, **semantic annotation** of the extracted data set ([2], [7]), **combination** [12] of obtained data sets, **filtering** of data sets in order to remove inconsistent or duplicated data. We divide each of these tasks as RESTful resources in our architecture and orchestrate them as workflows. In order to handle data flows between resources, we define an orchestrator, which acts as a data bus. This orchestrator forwards messages and data from and to resources, builds HTTP request and handles requests responses. Fig. 1 shows the different resources and components of our architecture. On top of that, we provide our architecture with management API and resources to avoid manual configurations as much as possible.

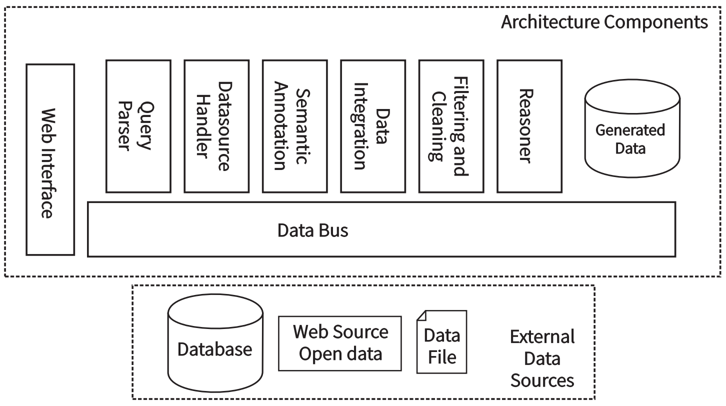


Fig. 1. Our Resource Oriented Architecture

3.2 Handling Volume Diversity

In order to handle volume diversity in our RESTful architecture, and to maintain a good response time performance, we propose a solution which reduces data transfer to a required minimum.

In our architecture, upon query request, an adapted workflow is generated, involving resources and services, to handle the required processing tasks. This workflow is executed, our data bus handles data exchanges, generating HTTP requests and retrieving responses from resources. We identified the following data transfers during the process: extraction from data sources, data transfer between core resources, and data download by the user. In this context, data extraction and download are required, but transfer between resources can be avoided or modified.

Based on this observation, we modify our architecture to decrease the volume of exchange data between resources. We design our API to only manipulate queries and metadata (data models, etc.). Computing this metadata, remote processing codes are generated in order to complete the process managed by this resource. These processes are handled close to data, minimizing execution time by lowering latency and network time.

3.3 Storing data

In order to *temporarily* store data, our proposal is based on temporary data storage units, generated at each user request. Storage units act as file hosting services, they are generated at runtime, for each *new* user request, for a limited amount of time and contains the data and metadata for this request. They are provided with an engine capable of processing remote processing tasks generated by resources. Storage units are erased after a certain amount time, which has been fixed to a default value of 24 hours. This delay is customizable for each request. Time counter is reset each time a user reissues the query or reaccesses the storage. Storage units are accessible as RESTful resources, for management purpose or to retrieve query responses when data processing is over. When the processing tasks are over, the storage URI is given to the user, so that he could download the data sets answering to his request.

3.4 Processing engine

In order to handle the different tasks required to complete the smart data process, we provide the storages with a functional engine capable of executing remote processing tasks directly above the data instances. These codes are generated by the tasks resources and transfered instead of data.

We rely on functional languages to define processing tasks, since our data sets are stored as tabular data sets (lists or dictionary in JSON or JSON-LD[10]). We provide data store with different processing engines, each representing an environment or an engine. Each data store is provided with an API, which allows its management, but also to register engine libraries or plugins, required to process the different langages or functionnalities This data store is provided with an API, allowing to register different engine libraries or plugin to handle specific languages or functionnalities.

4 Implementation and Evaluation

In this paper, we focus our work on handling data volume diversity in our architecture for smart data management.

In order to evaluate the scalability of our architecture, we demonstrate the evolution of performance, evaluating request response time when answering to a set of complex semantic queries over multiple data sources. We vary the number of data sources and measure response times.

4.1 Use Case Scenario

We focus our work on the enrichment and reusability of data, and based our models and implementation on the data handled by a communication company, which has a need for an adaptive system able to automatically refine and combine data coming from their internal information system and enrich it with help from open Web data sources. This solution has for objectives to study the impact of campaign broadcasts over a list of customers and to provide decision support tools for future broadcasts. This scenario describes the following data sources, presenting several characteristics, specific to our scenario:

Source 1 is a linked service giving access to our company small business data set. Data that come from this source are subject to privacy constraints. **Source 2** is a SQL endpoint to a database that contains millions of tuples with a low update frequency. **Source 3** is another SQL endpoint to a database that contains customer daily activities, updated regularly. **Source 4** is a RSS stream that contains user update requests (removal requests from customers, request form architecture's user). Other **sources** are represented by a set of Web sources: *FOAF* and *vcard* ontologies that help to annotate data, as well as a Dbpedia SPARQL endpoint. Relying on the scenario presented above, we create two request, involving different concepts subgraphs. We populate our scenario with a set of data sources, covering the different subgraphs.

```
PREFIX al: <http://restful.alabs.io/concepts/0.1/>
PREFIX xsd: <http://www.w3.org/TR/xmlschema-2/>
SELECT ?email_value ?campaign WHERE {
    ?email  a  al:email ;
            al:has_email_value  ?email_value ;
            al:blacklist_status ?status .
    ?clic  a  al:clic ;
           al:clic_email       ?email ;
           al:clic_date        ?date .
    FILTER  (?status != 1 && ?date >= "1411477450"^^xsd:date)
```

Listing 1.1. Query 1.1 involving the different concepts

```
PREFIX al: <http://restful.alabs.io/concepts/0.1/>
SELECT ?email_value WHERE {
    ?email      a                   al:email ;
        al:has_email_value  ?email_value ;
        al:blacklist_status ?status .
    FILTER  (?status != 1)
}
```

Listing 1.2. Query 1.2 introducing a user specific filters

Query 1.1 involves four subgraphs, implying data sources with different characteristics such as high volume (scenario's big database) and privacy sensitive information (scenario linked service). This query also presents user specific filters. Query 1.2 involves only a few number of concepts, and present less data manipulations. Tests are performed on a double core 2.3 GHz machine, with 4 GB of RAM. Restful resources and architecture are implemented through PHP frameworks, Zend and Slim[2] and hosted on scenario company servers (Apache).

4.2 Evaluation

We evaluate our architecture response time to query $Q1$, and $Q2$ respectively, when the number of data sources increases. We compared response time evolution for the same queries and data, with and without our optimisation process.

As can be seen in Fig. 2 and Fig. 3, the volume of data increase with the number of data source, and the classical approach response time suffer, due to HTTP transfers. In parallel, optimized approach response time increase linearly, but stays acceptable standing under the treshold of 4 seconds. In this case, transfer and network latency consumes time exponentially. Query $Q2$ involves less concepts, and as a result less data manipulations, but our architecture still suffer from a very high latency due to data transfer. When the number of data source exceed 24, the classical approach is unable to provide a response in an acceptable delay. This graph clearly show that our architecture can adapt to large volumes of data, using our optimisation technique.

[2] See http://www.slimframework.com/

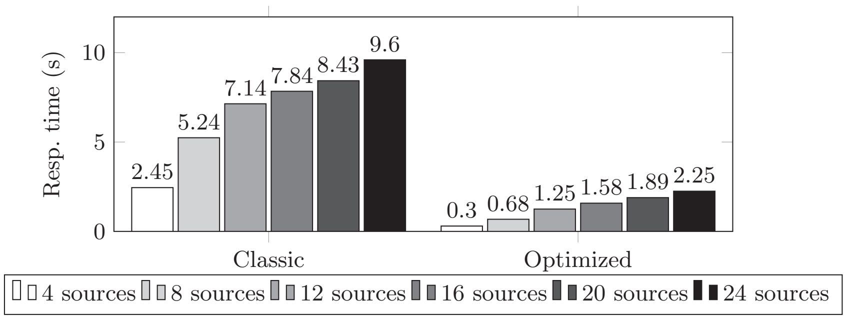


Fig. 2. Evaluation of average response time for query 1.1

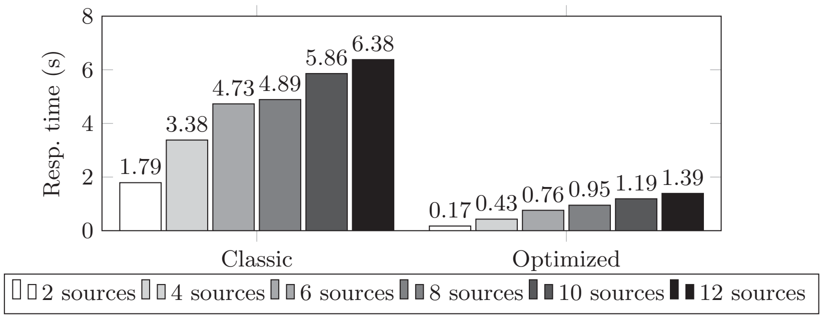


Fig. 3. Evaluation of average response time for query 1.2

5 Conclusion

In this paper, we describe a resource-oriented architecture that relies on adaptive data processing strategies to optimize data exchanges between resources that process data. We focus on the latency problems that appear when dealing with diverse data volumes especially when transferring data between the different architecture components. All along the smart data construction process, we rely on a temporary data storage where business data is stored. Our architecture handles the communication between the different resources, by transferring metadata about the query and the data storage URI. Resources generate adapted remote processing codes, which are forwarded to storage units and executed on-site by the data storage engine. Therefore, data manipulation is performed on-site, and the data does not flow through the architecture. By reducing latency due to data transfers, we alleviate our decentralized and distributed architecture from the burden of data tranfer, and improve the responsiveness of data-driven resource workflows. We support our implementation with a set of evaluations and tests through our use case scenario. As future work, we envision to investigate the appearance of data uncertainty in data aggregating approach, relying on probabilistic integration techniques.

References

1. Djamal Benslimane, Schahram Dustdar, and Amit P. Sheth. Services mashups: The new generation of web applications. *IEEE Internet Computing*, 12(5):13–15, 2008.
2. Christian Bizer. D2rq - treating non-rdf databases as virtual rdf graphs. In *In Proceedings of the 3rd International Semantic Web Conference (ISWC2004*, 2004.
3. Pierre De Vettor, Michaël Mrissa, and Djamal Benslimane. Models and architecture for smart data management. Technical Report RR-LIRIS-2014-017, LIRIS UMR 5205 CNRS/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon, December 2014.
4. Adrien Devresse and Fabrizio Furano. Efficient HTTP based I/O on very large datasets for high performance computing with the libdavix library. *CoRR*, abs/1410.4168, 2014.
5. A Ford, C Raiciu, M Handley, S Barre, and J Iyengar. Architectural guidelines for multipath tcp development. In *IETF, RFC 6182*. Citeseer, 2011.
6. Robert L. Grossman, Yunhong Gu, Xinwei Hong, Antony Antony, Johan Blom, Freek Dijkstra, and Cees de Laat. Teraflows over gigabit {WANs} with {UDT}. *Future Generation Computer Systems*, 21(4):501 – 513, 2005. High-Speed Networks and Services for Data-Intensive Grids: the DataTAG Project.
7. Lushan Han, Tim Finin, Cynthia Parr, Joel Sachs, and Anupam Joshi. Rdf123: From spreadsheets to rdf. In *The Semantic Web - ISWC 2008*, volume 5318 of *Lecture Notes in Computer Science*, pages 451–466. Springer Berlin Heidelberg, 2008.
8. Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2011.
9. R.K.L. Ko, M. Kirchberg, Bu-Sung Lee, and E. Chew. Overcoming large data transfer bottlenecks in restful service orchestrations. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 654–656, June 2012.
10. Markus Lanthaler and Christian Gutl. On using json-ld to create evolvable restful services. In *Proceedings of the Third International Workshop on RESTful Design*, WS-REST '12, pages 25–32, New York, NY, USA, 2012. ACM.
11. M. Maleshkova, C. Pedrinaci, and J. Domingue. Investigating web apis on the world wide web. In *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, pages 107–114, Dec 2010.
12. Michael Mrissa, Mohamed Sellami, Pierre De Vettor, Djamal Benslimane, and Bruno Defude. A decentralized mediation-as-a-service architecture for service composition. *2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 0:80–85, 2013.
13. Amit Sheth. Transforming big data into smart data: Deriving value via harnessing volume, variety, and velocity using semantic techniques and technologies. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 2–2, March 2014.
14. R. van der Pol, S. Boele, F. Dijkstra, A. Barczyk, G. van Malenstein, J. H. Chen, and J. Mambretti. Multipathing with mptcp and openflow. *High Performance Computing, Networking Storage and Analysis, SC Companion:*, 0:1617–1624, 2012.
15. Zibin Zheng, Jieming Zhu, and M.R. Lyu. Service-generated big data and big data-as-a-service: An overview. In *Big Data (BigData Congress), 2013 IEEE International Congress on*, pages 403–410, June 2013.

Relational XES: Data Management for Process Mining

B.F. van Dongen and Sh. Shabani

Department of Mathematics and Computer Science,
Eindhoven University of Technology, The Netherlands.
B.F.v.Dongen, S.Shabaninejad@tue.nl

Abstract. Information systems log data during the execution of business processes in so called "event logs". Process mining aims to improve business processes by extracting knowledge from event logs. Currently, the de-facto standard for storing and managing event data, XES, is tailored towards sequential access of this data. Handling more and more data in process mining applications is an important challenge and there is a need for standardized ways of storing and processing event data in the large.
In this paper, we first discuss several solutions to address the "big data" problem in process mining. We present a new framework for dealing with large event logs using a relational data model which is backwards compatible with XES. This framework, called Relational XES, provides buffered, random access to events resulting in a reduction of memory usage and we present experiments with existing process mining applications to show how this framework trades memory for CPU time.

1 Introduction

Over the last few years, the amount of historical execution data stored by large-scale information systems has grown exponentially. This data is typically stored for analysis purposes, for example to enable auditing, process improvement or process mining. The evolution of information system into such large-scale systems increases the need for business process analysis techniques to also handle data at such a large scale.

Most process mining techniques that exist today focus on the analysis of so-called *event logs* with the purpose to discover, monitor and improve business processes. Traditionally, process mining techniques assume that it is possible to *sequentially* record *events* into *traces* such that each event refers to an *activity* (i.e., a well-defined step in the process) and is related to a particular case (i.e., a process instance). Event logs may store additional information such as the *resource* (i.e., person or device) executing or initiating an activity, the *timestamp* of an event, or *data elements* recorded with an event (e.g., the size of an order).

This classic sequential view on a log data has a few important downsides. First, each event is assumed to be relevant for only one trace in the context of one process. In real life this is not necessarily the case, i.e. events may be relevant in the context of many different traces belonging to different processes. Letting an event appear in many traces requires a significant amount of duplication. Second, the tree structure of an event log is great for sequential access to the log, but not suitable for random access, while most techniques actually use a random access paradigm to access events in the log. Third, the current rise of decomposition and distribution based techniques for process mining requires easy filtering of event logs both vertically, i.e. distributing cases over different logs and horizontally, i.e. distributing events over different logs.

In this paper, we present an architecture that addresses the issues above while it maintains backwards compatibility with existing process mining techniques. Our architecture uses a database to store the event log, allowing for events to appear in multiple traces, for events to be accessed in a random-access fashion and for efficient filtering.

The remainder of this paper is structured as follows. In Section 2 we discuss related work. Then, in Section 3, we present our framework for storing and managing event data. In Section 4 we compare our framework with the de-facto standard for managing event data and we conclude the paper in Section 5.

2 Related Work

Process mining is a research discipline that provides techniques to discover, monitor and improve processes based on event data. It is beyond the scope of this paper to give a full introduction to process mining, but we refer to [7] for such an overview.

Most process mining techniques today have been assuming that event data is available in the form of sequential traces of ordered events that are typically ordered in time. More recently however, process mining researchers have come to realize that events are typically related to multiple processes being executed in parallel while synchronizing on some activities [4–6]. Furthermore, research is emerging on the analysis of streams of events where the notion of a trace is not predefined but may change over time. [1, 3]

Since the start of research on process mining, attempts have been made on standardizing the way in which event data is to be stored. This has lead to a number of semi-standards, such as MXML [9] which was a simple XML format for audit trails of process aware information systems and the recent XES format [10, 11]. The XES format is supported by both academic tools such as ProM [11] as well as industrial tools such as Disco [2]. Many real-life datasets have been made public in the XES xml format

OpenXES is a reference implementation for dealing with event data in the XES format. This reference implementation consists of a number of interfaces as well as a number of concrete implementations. These interfaces allow for both sequential and random, read and write access to event data. Over the years, this implementation has proven to be very successful in the open source process mining framework ProM [11]. Most implementations of OpenXES keep entire logs in memory, while others use internal databases. However, all implementations use XML as their serialization format.

It is not the first time that databases are used to store event logs. XESAme [11] is a log-import framework that allows events from other systems to be converted to a XES file. Internally, this framework uses a temporary database to store event data, but then this database is not exposed to the user and the contents are always serialized to the XES XML format.

3 Relational XES (RXES)

XES is an open standard for *storing* and *managing* event data. For the purpose of storing event data, a standardized, extensible storage format was developed, of which the definition is shown in Figure 1. In XES, each event, trace and log is annotated with typed or untyped attributes which are given semantics through so-called extensions. For example, the activity an event refers to is stored as a literal attribute with key "concept:name". This key is defined in the standard "Concept Extension" [10, 11].

One of the fundamental assumptions of XES is that each event belongs to exactly *one* trace and occurred in the context of exactly *one* log as shown in Figure 1. In practice however, this is often not the case [4].

Therefore, in this paper, we present the RXES framework that lifts this assumption. Instead of considering an event to have occurred in the context of a particular trace, we consider a trace to be a collection of events and a log to be a collection of such traces, but events may occur in many different traces and traces may appear in multiple logs. This allows for backwards compatibility with traditional mining techniques that rely on the "single trace" view, while also allowing for more advanced techniques to consider multiple views at once without the need for duplicating these events.

In Figure 2 we show an ER diagram for the RXES framework. The framework uses a scheme to store events in a database that is largely based on the UML class diagram used for XES but separates the contents of events and traces from their appearance in traces and logs respectively, thus requiring significantly less data duplication. In the remainder of this section, we discuss the main differences between XES and RXES.

3.1 Representation of events and traces

In RXES, logs, traces and events are represented by tables with ids. The actual state of an `XTEvent` is dependent on values of its attributes (accessed through the `XAttributable` class). Similarly, an `XTrace` is nothing more than an ordered list of events (represented by the composition) which carries state through its attributes and can only exist in the context of a log. Therefore, there is no need to have content in the tables representing traces and events in RXES.

Events occurring in the context of a particular trace are represented by the `trace_has_event` table which identifies *occurrences* of events rather than the events themselves and similarly *trace occurrences* are represented by the `log_has_trace` table. As events can appear in multiple traces at different locations and traces can appear in multiple logs at different locations, we keep an order number indicated by `sequence` that is specific to the occurrence of an event in a trace or a trace in a log.

An event appearing in multiple traces can in RXES be represented by a single entry in the event table, but the schema is flexible enough to allow for duplication of the event for each occurrence if desired.

3.2 Representation of Attributes

In XES, attributes are represented by a single class `XAttribute`. Each attribute is composed of a single key and is typed through one of its subclasses (Boolean, Con-

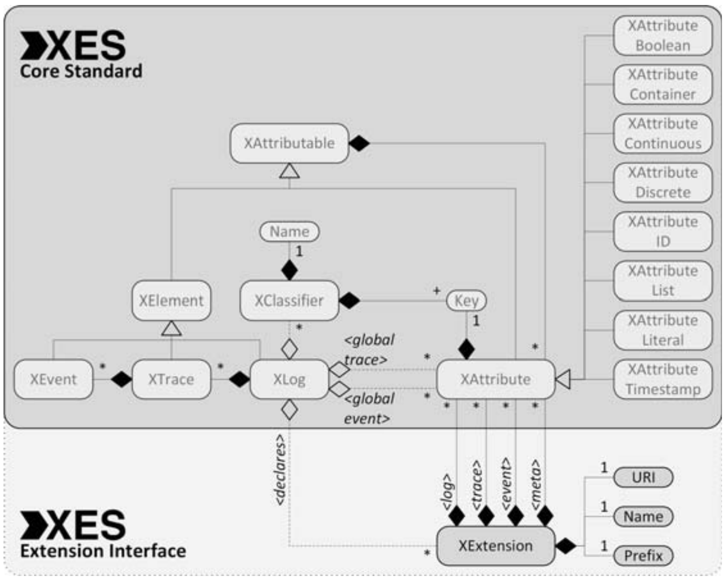


Fig. 1: UML class diagram of the XES standard [10, 11]

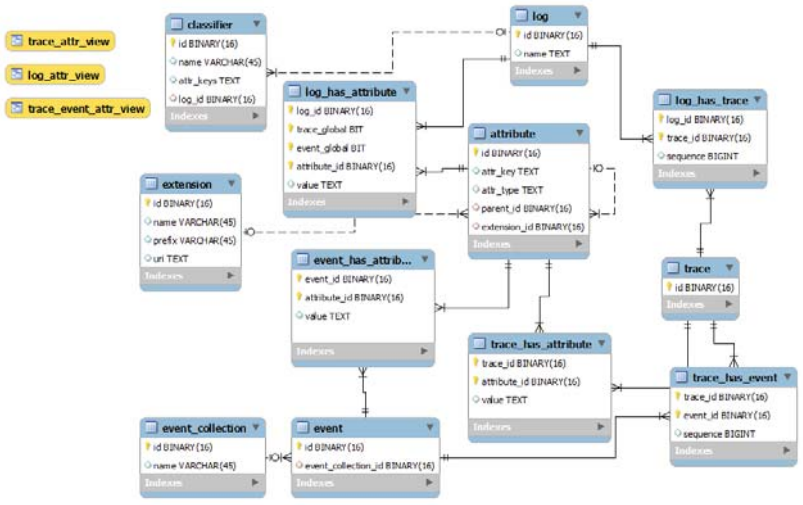


Fig. 2: ER Diagram for RelationalXES

tainer . . . Timestamp). The `XAttribute` class may carry attributes itself (i.e. meta-attributes). In RXES, attributes are separated from values to reduce data duplication and to enforce uniqueness of an attribute's type. The latter is technically not a requirement in XES, but it is considered good practice not to use the same attribute with more types. To cover meta attributes the `attribute` table includes a parent child relationship expressing that attributes have attributes. The values of the meta attributes are stored in the `event has attribute` table, hence the definition of a meta attribute that is contained in more than one event attribute with a different value is stored for each value. This is a design choice to avoid having the relate events occurrences with attributes.

As shown in Figure 1, event logs may contain a number of global attributes. The notion of global events refers to the idea that a log can specify that a particular attribute is present on all traces (or events) contained within it, i.e. the attribute is defined to be globally present. This allows for process mining techniques to verify whether a log satisfies certain input conditions, such as the presence of timestamps. A global attribute also specifies a default value which can be used for example when adding new traces or events to a log or in case an attribute declared to be global is nonetheless not present. In RXES, we use two binary attribute in the `log_has_attribute` table to indicate if an attribute is global. To cover the concept of global attribute it has been enforced by the schema that there should not be two global attributes in one log with the same key but a different value.

3.3 Representation of Classifiers and Extensions

Another feature of XES is the availability of so-called classifiers and extensions which provide semantics to events in a log. A classifier in XES is composed of a collection of attribute keys. If a classifier is included in a log, it provides insights into the way individual events should be translated into business activities (or event classes in XES terms). The idea is that such classifiers provide a starting point for process mining techniques to reason on the correct level of abstraction. In practice, classifiers are rarely contained in a log and are often added by the process mining technique. Therefore, in our database, we represent the attribute keys of classifiers simply by text. As a general rule, a classifier should only refer to keys of event global attributes, but this is not enforced by XES nor by RXES.

The last concept of XES that is supported by RXES are the so-called extensions. The extension mechanism allows users to give semantics to attributes. These extensions specify specific keys for attributes which have to be interpreted in a standardized way. For example, the concept extension defined by [10] defines attributes `concept:name` of type `String` which *stores a generally understood name for any type hierarchy element. [...] e.g. the name of the executed activity represented by the event*[10]. In RXES, extensions are stored in a separate table, and attributes are connected to extensions using foreign keys.

3.4 Identification of attributes

The attribute table uses auto-generated IDs attached to each attribute to connect attributes of the same type using SQL queries. By using an id as a primary key rather than other fields, we allow for recognition of identical attributes which is quite useful for decomposition, filtering and importing. When encountering an attribute that exists in the whole log, we may add a reference to the existing id in the database using an in-memory cache of existing attributes. However, if an attribute is not in cache, we add the attribute using a fresh id. The consolidation of identical attributes can be done offline.

4 Benefits of RelationalXES over OpenXES

RelationalXES provides a full implementation of all OpenXES interfaces using the database as a backend. As a result the framework is fully backward compatible and provides a number of benefits over OpenXES. First, keeping events in the database and only retrieving minimal amounts of data per each request, reduces the memory usage of process mining techniques significantly. Second, through SQL-based filtering and database views, decomposition algorithms no longer require event data to be duplicated in memory, and third, events can exist outside of the context of a trace or a log, allowing RXES to store event data from streams and to utilize trace identification techniques.

RXES has been implemented in Java ,and for the experiments in this paper, MySQL has been used as a database system. However, the design of the application allows for any relational database systems. The framework keeps only the data items that are directly related to the log entity plus list of ids of traces in the memory and later loads the actual data if it is necessary, i.e. if it is requested by a process mining technique. RXES is implemented as a package in the ProM framework and is available as an open-source implementation. It can be downloaded from http://www.processmining.org/.

Currently OpenXES has multiple implementations included in ProM. In this section, we show the difference between our new technique and existing implementations by evaluating memory and CPU usage of processing different sizes of logs.

To investigate memory and CPU usage we used 30 sample event-logs. For that, we used a public, real-life dataset [8] as a base. This dataset contains 13,087 traces with 262,200 events. In total, there are 1,082,719 attributes contained in these events. To investigate the behavior of the system, we extend the size of this log in different dimensions, i.e., we increased the number of events and attributes of the base log up to 10 times . We used a binary search technique to find the least amount of memory needed to process each event log. The resulting memory use for each log and each implementation is shown in Figure 3(a) and Figure 3(b).

Clearly, the default OpenXES implementation (using Java's Collection Framework) has the highest memory allocation. The existing MapDB implementation (in package XESLite) drops the usage by 90%, while our technique uses 90% less than that. For RXES, these results include the memory needed by the Database Management System.

Figures 3(c) and 3(d) show a comparison for the time needed to access all events in the event logs. To evaluate time, a program that requests all elements of the event log has been executed 10 times and an average time is computed. The amount of memory given to the Java virtual machine is sufficient to meet the demands of the Default OpenXES implementation. Clearly, there is a direct relation between memory usage and speed of access. As Figure 3(c) shows, the default implementation is the fastest method simply because the event log is loaded into memory completely. The figures further show that

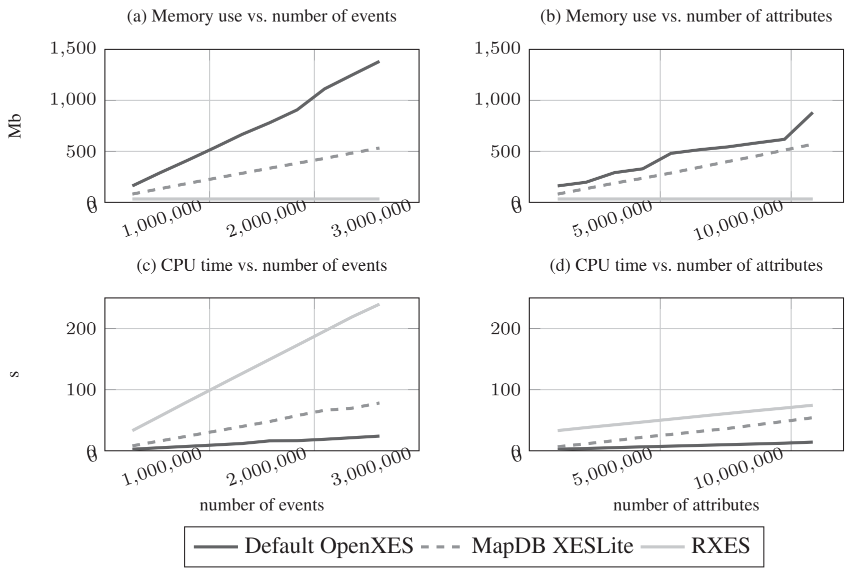


Fig. 3: Memory and CPU use for accessing all events in a log vs. number of events and attributed in the log.

RXES is slower than MapDB which uses an in-memory database, which is mostly because for this experiment we used minimum size of the trace buffer which only keeps one trace at a time. Hence, the execution time is result of performing many SQL queries over a TCP/IP connection to the DBMS.

In general, if the size of the event log is small compared to the available memory, MapDBDisk keeps a good balance between memory and time usage. However, compared to RXES, MapDB doesn't allow for distribution and replication, while it is vital in cases of using decomposition and parallelism to handle the big data.

The ability use of RXES to process large event logs with readily available process mining techniques is essential for the adoption of our framework in practice. However, our framework is capable of performing tasks directly on the database, such as event log filtering.

5 Conclusion and Future Work

In this paper, we presented RelationalXES. This framework is a generalization of the de-facto standard XES for storing and managing event data in process mining. Where all existing implementations of XES uses the strict notion of containment for events in traces and traces in logs, our framework is much more flexible and allows for events to appear in more than one trace and for traces to appear in more than one log. That makes it possible to have different views on the same event log. Furthermore, the database schema used in RXES allows for a significant reduction of duplication by storing frequently occurring attributes only once rather than repeating them for every occurrence.

In the paper, we presented the framework in detail and we discuss the underlying database schema. Furthermore, we show the reduction of memory use by process mining techniques using RXES.

References

[1] Andrea Burattin, Alessandro Sperduti, and Wil M. P. van der Aalst. Heuristics miners for streaming event data. *CoRR*, abs/1212.6383, 2012.

[2] Christian W. Günther and Anne Rozinat. Disco: Discover your processes. In Niels Lohmann and Simon Moser, editors, *BPM (Demos)*, volume 940 of *CEUR Workshop Proceedings*, pages 40–44. CEUR-WS.org, 2012.

[3] Fabrizio Maria Maggi, Andrea Burattin, Marta Cimitile, and Alessandro Sperduti. Online process discovery to detect concept drifts in ltl-based declarative process models. In Robert Meersman et.al.s, editor, *OTM Conferences*, volume 8185 of *Lecture Notes in Computer Science*, pages 94–111. Springer, 2013. ISBN 978-3-642-41029-1.

[4] Erik H. J. Nooijen, Boudewijn F. van Dongen, and Dirk Fahland. Automatic discovery of data-centric and artifact-centric processes. In Marcello La Rosa and Pnina Soffer, editors, *Business Process Management Workshops*, volume 132 of *Lecture Notes in Business Information Processing*, pages 316–327. Springer, 2012. ISBN 978-3-642-36284-2.

[5] Viara Popova and Marlon Dumas. Discovering unbounded synchronization conditions in artifact-centric process models. In Niels Lohmann, Minseok Song, and Petia Wohed, editors, *Business Process Management Workshops*, volume 171 of *Lecture Notes in Business Information Processing*, pages 28–40. Springer, 2013. ISBN 978-3-319-06256-3.

[6] Viara Popova, Dirk Fahland, and Marlon Dumas. Artifact lifecycle discovery. *CoRR*, abs/1303.2554, 2013.

[7] Wil M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011. ISBN 978-3-642-19344-6.

[8] B.F.; van Dongen. Bpi challenge 2012, 2012. URL `http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f`.

[9] Boudewijn F. van Dongen and Wil M. P. van der Aalst. Emit: A process mining tool. In Jordi Cortadella and Wolfgang Reisig, editors, *ICATPN*, volume 3099 of *Lecture Notes in Computer Science*, pages 454–463. Springer, 2004. ISBN 3-540-22236-7.

[10] H. M. W. Verbeek and Christian W. Günther. XES standard definition 2.0. Technical report, BPMcenter.org, July 2014. URL `http://bpmcenter.org/wp-content/uploads/reports/2014/BPM-14-09.pdf`. BPM Center Report BPM-14-09.

[11] H. M. W. Verbeek, Joos C. A. M. Buijs, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. XES, XESame, and ProM 6. In Pnina Soffer and Erik Proper, editors, *CAiSE Forum*, volume 72 of *Lecture Notes in Business Information Processing*, pages 60–75. Springer, 2010. ISBN 978-3-642-17721-7.

Specifying Business Requirements through Interaction Design

Roman Popp and Hermann Kaindl

Vienna University of Technology, Gusshausstr. 27–29, A–1040 Vienna, Austria

{popp, kaindl}@ict.tuwien.ac.at

Abstract. When the requirements and the interaction design of a system are separated, they will most likely not fit together, and the resulting system will be less than optimal. Even if all the real needs are covered in the requirements and also implemented, errors may be induced by human-computer interaction through a bad interaction design and its resulting user interface. Such a system may even not be used at all. Alternatively, a great user interface of a system with features that are not required will not be very useful as well.

This tool demo illustrates joint modeling of (communicative) interaction design and requirements, through a Discourse-based Communication Model consisting of a Discourse Model, a Domain-of-Discourse Model and an Action-notification Model. The concept of such a Discourse Model was derived from results of human communication theories, cognitive science and sociology (even without employing speech or natural language). While these models were originally devised for capturing interaction design, it turned out that they can be also viewed as specifying classes of scenarios, i.e., use cases. In this sense, they can also be utilized for specifying requirements. Ontologies are used to define the domain of discourse for the interactions and the actions that can be performed through the interactions with a software system. These models specify both the business requirements and the interaction design.

User interfaces for software systems can be generated semi-automatically from our Discourse Models, Domain-of-Discourse models and Action-Notification Models. This is especially useful when user interfaces for different devices are needed, since graphical user interfaces (GUIs) for PCs will most likely not fit relatively small screens of devices like today's smartphones. Based on an additional device specification, our tool automatically tailors the generated GUI to the given device.

So, interaction design facilitates requirements engineering to make applications both more useful and usable.

Keywords: Business requirements, Scenarios, Interaction Design, Discourse Models, Generation of multi-device GUIs

References

[1] Bogdan, C., Kaindl, H., Falb, J., and Popp, R., Modeling of interaction design by end users through discourse modeling, in Proc. of IUI'08, 2008.

[2] Falb, J., Kaindl, H., Horacek, H., Bogdan, C., Popp, R., Arnautovic, E., A discourse model for interaction design based on theories of human communication. In: CHI'06 Ext. Abstracts on Human Factors in Computing Systems, pp 754–759 ACM Press (2006)

[3] Kaindl, H., Constantine, L., Pastor, O., Sutcliffe and A., Zowghi, D. How to Combine Requirements Engineering and Interaction Design?. In RE'08, 2008.

[4] Kaindl, H., Popp, R., and Raneburger, D., Automated Generation of User Interfaces: Based on Use Case or Interaction Design Specifications?. In: ICSOFT'12.

[5] Popp, R., Raneburger, D., and Kaindl, H., Tool Support for Automated Multi-device GUI Generation from Discourse-based Communication Models. In: EICS'13, 2013.

[6] Raneburger, D., Kaindl, H., and Popp, R. Strategies for automated GUI tailoring for multiple device. In: HICSS-48, 2015.

[7] Raneburger, D., Popp, R., Kavaldjian, S., Kaindl, H., Falb, J., Optimized GUI Generation for Small Screens, Model-Driven Development of Advanced User Interfaces, LNCS, Springer-Verlag, Berlin-Heidelberg (2011)

On Business Process Variants Generation

Asef Pourmasoumi[a,b], Mohsen Kahani[b], Ebrahim Bagheri[a], Mohsen Asadi[c]

[a] Department of Electrical and Computer Engineering, Ryerson University, Canada
[b] Web Technology Lab, Ferdowsi University of Mashhad, Iran
[c] SAP Canada, Vancouver Canada
a.pourmasoumi@ryerson.ca, kahani@um.ac.ir, bagheri@ryerson.ca, masadi@sfu.ca

Abstract. Cross-organizational mining is a new research field in the process mining domain, which focuses on the analysis and mining of processes in multiple organizations. Suitable access to collections of business process variants is necessary for researchers to evaluate their work in this research domain. To the best of our knowledge, no complete collection of process variants or any process variants/log generator tool exists for this purpose. In this paper, we propose an algorithm for generating random process variants for a given process model and a supporting toolset built on top of the PLG toolset. For this purpose, we classify different factors that can serve as variation points. Then, using the structure tree based representation of an input process, we present an algorithm for applying variation points based on a user-defined variation rate. The developed tool is publicly available for researchers to use.

Keywords: process model variants, process variant generator, variation point, Process structure tree

1 Introduction

Peer-organizations such as municipalities, hospitals and universities often employ many different variations of the same business processes. For instance, Suncorp is a famous Australian insurance company, which has over 6000 business process variants [1]. These processes have many commonalities and some degree of variability. Mining and analysis of such process variants can result in insights that can improve organization operations [2]. Based on the literature [3] [4] [5], variants are defined as process models, which follow the same goals but have slight structural differences, i.e. they have at least one feature in common and one feature in which they differ.

Cross-organizational mining encompasses different research branches: reference model extraction, process models similarity calculation, process model merging, process fragmentation, among others. In all of these research areas, there is dire need for collections of process variants and/or execution logs. Unfortunately there are not enough standard datasets of process variants or variants executions logs. The only available dataset is a small collection of process variants log in the BPI Challenge2014, but it just contains log files of five variants of a process model from the

CoSeLOG1 project. The lack of public data can be attributed to disinclination of organizations to publish their own data. So, simulation and process synthesis tools can serve as a viable alternative for researchers for the evaluation of their techniques.

In this paper, we present an approach and toolset for generating process variants. We provide the following contributions:

- First, we classify the effective factors for creating process variants. These factors can be used as a reference for proposing new algorithms and tools for process variant generation.
- Second, we propose an algorithm based on the structure tree representation of input process models for creating variation points on an input process model. Using this algorithm, a collection of process variants can be created randomly according to a probabilistic distribution and based on a user-defined variation rate parameter.

2 Related Works

Cross-organizational mining is a young research field in the process mining domain [6]. The prerequisite for research in this sub-domain is having appropriate process variants. Large collections of process variants enable researchers to comprehensively evaluate their work. In the past, there have been several works on random process generation. In [7], the PLG (process log generator) tool for generating random block-structured process models and their executions is presented. In the most of references, block-structured process models require that each control flow split has a corresponding join of the same types [8][9]. This tool is open source and has a plug-in for the ProM framework. PLG generates random process models using context-free grammars by employing 5 basic workflow patterns: single activity, loop, sequence, XOR split-join and AND split-join. Moreover, the user can select from three probability distribution functions: Uniform, Gaussian and Beta, which will be used for generating the number of branches for AND/XOR split-join patterns. After generating a random process model, PLG is capable of generating its execution logs by traversing the generated process graph. PLG is a great framework for generating large scale random process models and event logs. Hence, we used it as the basis for developing our process variants generator tool.

There are also other tools for generating process models [10] [11]. In [10], CPN toolset is proposed. It has a powerful GUI and users can easily edit process models. An extension of CPN is capable of generating event logs. However it uses a rare language for writing scripts which makes it difficult for development.

Shugurov et al. [12] propose an approach for generating a set of event logs with noise which is implemented as a plug-in for ProM. Their basic idea is to use token replay in Petri Net process models for log generation. They add noise using three ways: adding artificial transitions, adding existent transitions in incorrect order and by skipping events.

Groups	Function	Description
Operators Change Functions	$\Phi_{O_1 \mapsto O_2}$	This function converts a specified operator O_1 to operator O_2. The operators O_1 and O_2 can be: AND, XOR, OR, Sequence.
	$\Phi_{Delete}(O)$	This function deletes specified operator O.
	$\Phi_{Move}(O, A, B)$	This function moves a specified operator O to a new places between node A and B.
	$\Phi_{Add}(O, A, Childs)$	This function add a specified operator O before A and connects it to the list of Childs.
Activities Change Functions	Δ_{Add} (A, R_A)	This function inserts a new activity A with relation R_A to process.
	Δ_{Delete} (A)	This function removes the activity A.
	Δ_{Swap} (A, B)	This function swaps the activity A with the activity B.
	Δ_{Move} (A, C, D)	This function moves the activity A to a new place between C and D.
Connections Change Functions	Γ_{Add} (L_{AB})	This function adds a new sequence link from A to B.
	Γ_{Remove} (L_{AB})	This function delete a sequence A to B link.

The only work that we have encountered for generating process variants is proposed in [11]. Stocker et al. proposed SecSy for generating a set of event logs with some deviation from the original model.
The SecSy tool generates event logs for the sake of evaluating of business process security monitoring and auditing and is useful for security-oriented information systems [12]. One of the drawbacks of SecSy is that it does not cover many possible deviation patterns. It uses three transformations for making a deviation: Converting AND to XOR, XOR to AND and swapping the order of two activities, while there can be several other forms of transformation which will be explained in this paper. Also, in the SecSy tool, the user is not able to determine the degree of variation.

3 Proposed Approach

For the sake of clarity, we divide this section into three sub-sections; Section 3.1 introduces the various factors that can generate a process variant. These factors are classified into groups and can be used as a reference for proposing new algorithms for process variant generation. In Section 3.2, a structure tree based representation of BPMN process models is shown. In Section 3.3, we use this tree-based representation for extracting process variants. The proposed algorithm for extracting variants is described precisely in this section. Also, the probability distribution functions that have been used for random process variant generation is described in Section 3.3.

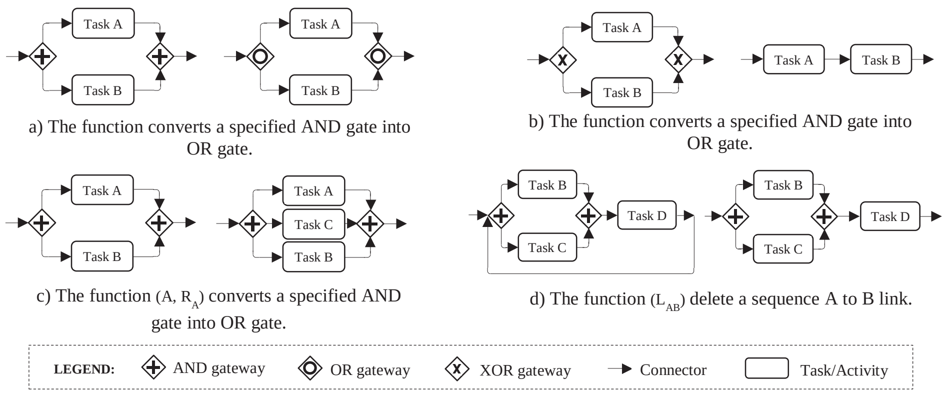

Task A
Task B
Task A
Task B
a) The function converts a specified AND gate into OR gate.
Task A
Task B
Task A
Task B
b) The function converts a specified AND gate into OR gate.
Task A
Task B
Task A
Task C
Task B
c) The function (A, R_A) converts a specified AND gate into OR gate.
Task B
Task C
Task D
Task B
Task C
Task D
d) The function (L_AB) delete a sequence A to B link.
LEGEND:
AND gateway
OR gateway
XOR gateway
Connector
Task/Activity

Fig. 1. BPMN representation of some functions described in Table 1

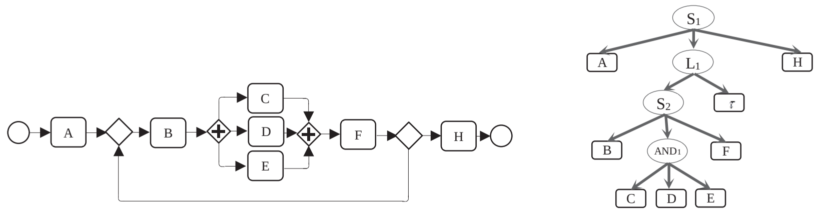
3.1 Factors for Variant Generation

In Table 1, we show the list of various functions that could result in process variants. This list is not intended to be comprehensive. The list is classified into three groups: *i*) Operator change functions, *ii*) Activity change functions and *iii*) Connection change functions. In the first class, the functions that lead to a variant using a change in operators of input process model are described. For example, $\Phi_{AND\mapsto OR}$ converts a given specified AND gateway to OR gateway and leads to a new process variant. As another example, the function $\Phi_{S\mapsto XOR}$ converts a given number of activities that have sequence relation and places an XOR gateway between them. The second class of functions changes the activities of the main process model for generating process variants. For example, the function Δ_{Insert} (A, R_A) inserts a new activity *A*, which did not exist in the main process model. The argument R_A determines the relations that *A* would have with other activities. Finally, the functions in the third class change the links that are in the main process model and lead to new process variants. For example, the function $\Gamma_{Remove}(L_{AB})$ deletes a specified link from activity *A* to activity *B*.

In Figure 1, the effects of some functions on an input process model are shown as an example. For the sake of simplicity, the process model is shown in BPMN format. It is clear that these functions have different magnitude of changes on the input process model.

3.2 Structure Tree Representation of Block-Structured Process Models

For implementing the functions in Table 1, in the first place, a representation for process models should be selected. In this paper, we use the structure tree representation of process models [13]. In selecting suitable representation, we consider two points: *i)* that the change functions in Table 1 can be applied directly, *ii)* that the representation can support block-structured process models. The reason for focusing on



a) An example BPMN process model b) Structure tree of process model (a)

Fig. 2. An example of the structure tree (S_i shows the sequence i and L_i shows Loop i)

block-structured processes is twofold: 1) In [13], translation from the widely used process modeling notations such as BPMN and Petri Net to structure tree and vice versa has been shown. 2) It is shown in [14] that about 95% of process models are block-structured or can be converted to an equivalent block-structured process.

In [14], a structure tree is defined as follows:

Definition 1 [14]: A process structure tree is a tuple T = (N, C, E; L) where:

- N is a set of leaf nodes representing activities.
- C is a set of connector nodes including AND, OR, XOR, Sequence, and Loop.
- $E \subseteq (C \times C) \cup (C \times N)$ is a set of edges.

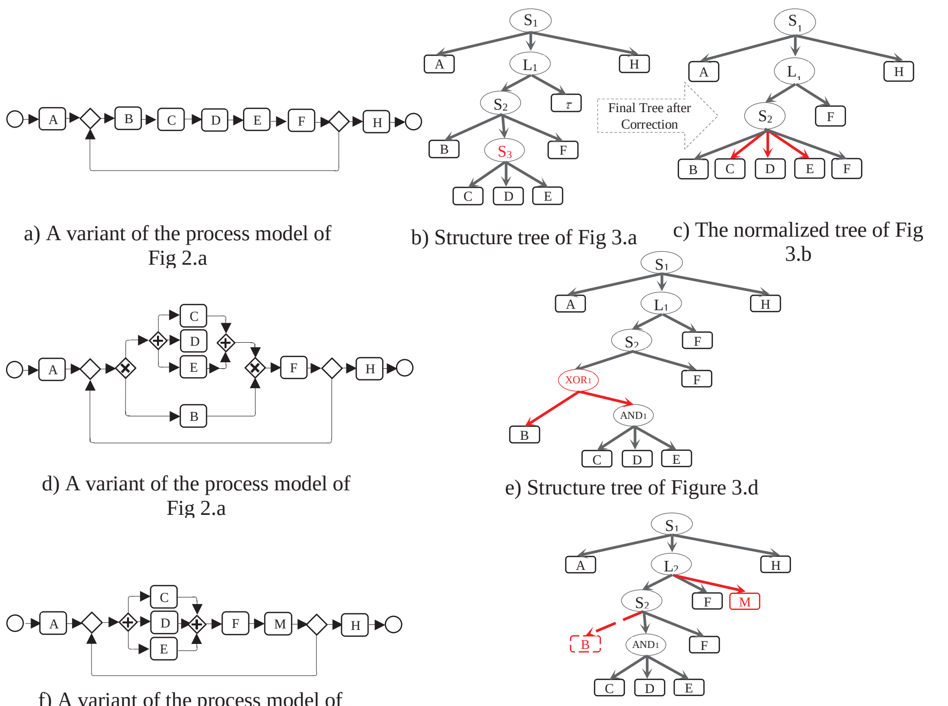
In Figure 2, an example of a block-structure process model and its corresponding structure tree is shown. In the structure tree, each intermediate node shows a process block and the tree is parsed from left to right. The intermediate nodes include AND-blocks, XOR-blocks, OR-blocks, Loops and Sequences corresponding to different patterns in process models. The leaf nodes in the tree correspond to activities in the process model.

Another reason for choosing structure tree as the representation form is that it can clearly show the blocks of a process model and their relationships. This would guarantee the soundness of the created variant after the change functions are applied. Since the changes are performed randomly and sequentially, so every permissible change applied on the process tree should keep the soundness of the process model. Moreover, every change in Table1 can be mapped to a change or a set of changes on the structure tree.

3.3 Generating Process Variants

The core of our idea for is to develop a mapping between the functions in Table 1 and change operations in the process structure tree. In other words, we intend to elicit the process variants by converting the structure tree of an input process model into other valid structure trees using tree conversion operations. These conversion operations will be selected based on the rate of variability, which the user specifies and through the mappings.

The operator change function ($\Phi_{O_1 \mapsto O_2}$) can be performed by changing the type of the connectors. For example, Figure 3.a shows a variation of the example in Figure



a) A variant of the process model of Fig 2.a

b) Structure tree of Fig 3.a

c) The normalized tree of Fig 3.b

d) A variant of the process model of Fig 2.a

e) Structure tree of Figure 3.d

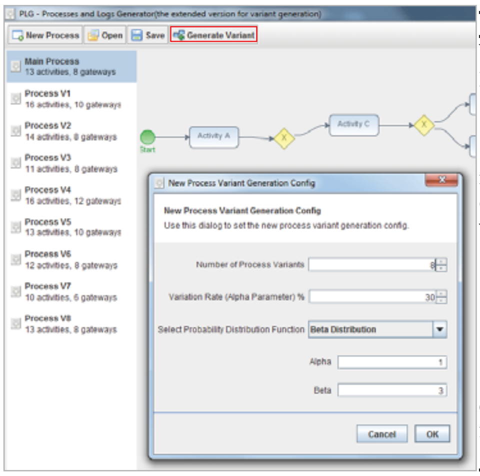
f) A variant of the process model of Fig 2.a (deleting B and adding M)

g) Structure tree of Figure 3.f

Fig. 3. Examples of mapping between change operations in the structure tree of Figure 2 and the functions in Table 1.

2.a. In this variation, AND gateway which sits between C, D and E has been changed to a sequence between C, D and E using the $\Phi_{AND_1 \mapsto S}$ function. The corresponding changes that need to be applied on the structure tree in order to enact this change are distinguished with red color. After every change, the modified tree would be checked and if there are any connector nodes that have at least one child of the same type (except for loop connector), the child node will be removed and its children will be connected to the parent.

The function $\Phi_{Delete}(O)$ can be implemented by removing the corresponding connector node and all of its children. The function $\Phi_{Move}(O,A,B)$ moves the block of operator O between A and B. For mapping this function on the structure tree, the parent of A would be checked. When the parent of A is OR/XOR/AND/Sequence, the subtree of O would become the right sibling of A. If the parent of A is a loop, given moving subtree of O after A requires the creation of a new connector node, we will do this by using the $\Phi_{Add}(O,A,Childs)$ change function. This function adds a new operator after A and connects it to the 'Childs' nodes. For mapping this function on the structure tree, we look at the children of A. If A has more than one child, we randomly select *n* consecutive children of A (where *n* is less than the number of children of A). Then A is connected to the newly generated operator O whereby O becomes the parent of the *n* selected children of A. It is also possible to add a new operator O with a new activity


PLG - Processes and Logs Generator(the extended version for variant generation)
New Process
Open
Save
Generate Variant
Start
Activity A
Activity C
X
X
Cancel
OK

Main Process
13 activities, 8 gateways

Process V1
16 activities, 10 gateways

Process V2
14 activities, 8 gateways

Process V3
11 activities, 8 gateways

Process V4
16 activities, 12 gateways

Process V5
13 activities, 10 gateways

Process V6
12 activities, 8 gateways

Process V7
10 activities, 6 gateways

Process V8
13 activities, 8 gateways

New Process Variant Generation Config

New Process Variant Generation Config
Use this dialog to set the new process variant generation config.

Number of Process Variants: 8
Variation Rate (Alpha Parameter) %: 30
Select Probability Distribution Function: Beta Distribution
Alpha: 1
Beta: 3

Algorithm1. The pseudo code of proposed approach

```
VariantGenerator($T$, $\alpha$, $d$, $p$)
Input: $T$ is structure tree representation
       $\alpha$ is variation rate,
       $d$ is probability distribution function,
       $p$ the parameter of selected distribution function
originalTree = T;
while(true) do
    f = selectVariationFunctionByProbabilityDF (d, p);
    if(treeBaseDistance(T,originalTree)> $\alpha$ ) then
        break;
    end
    T = performChangeOnTree(T, f);
    T = correctTree(T);
end
return T;
```

a) A screen shot of proposed tool. b) The algorithm of proposed approach

Fig. 4. A screen shot and algorithm of the proposed tool

child. In our tree-based mapping, all of these functions are necessary, and none of them can be implemented by other operation change functions.

The next group are activity change functions. The function $\Delta_{Add}(A, R_A)$ creates a new activity and connects it to existing connector nodes (adding a new activity to a new operator can be done using the $\Phi_{Add}(O, A, Childs)$ function). For mapping this function, there is need for creating a new activity and adding it randomly to existing connectors (Figure 3.f). The function $\Delta_{Delete}(A)$ removes activity A from the process model. It can be mapped to the tree be removing leaf node A. After removing activity A, its parent would be checked. If its parent is OR/XOR/AND/Sequence connector and has only one child, then its parent will be removed and the child is connected to its grandparents (Figure 3.g). The function $\Delta_{Move}(A, C, D)$ ~~is used for moving an activity~~ within a block or from one block to another block. This can be mapped in a tree by moving a leaf node between its siblings (when its parent is Sequence connector) or by changing its parent (when its parent is OR/XOR/AND). The function $\Delta_{Swap}(A, B)$ can be calculated using Δ_{Move}. Since our processes are sound and block-structured, we map $\Gamma_{Add}(L_{AB})$ and $\Gamma_{Remove}(L_{AB})$ functions just for adding and removing loops.

3.4 Selecting Generated Variation Based On The Variation Rate

The variation rate is a parameter that specifies the degree of permitted deviation from the input process model. We consider α as a variation rate, which will be determined by the user. In Table 1, various functions for variation creation have been listed. Each of these functions has a different effect on the input process model, but since these change functions are selected randomly and some of them may affect the previous changes (e.g. executing $\Delta_{Swap}(A, B)$ and $\Delta_{Swap}(B, A)$ does not make any change on the

input process), so in our tool, we exploit process similarity as introduced in [15] to measure degree of change. Pawlik and Augsten have introduced a tree-edit distance whereby the similarity of two trees is calculated based on the minimum number of operations that is needed to change one tree to another. In our work, we use the tree-edit distance for generating process variants such that the functions in Table 1 are chosen in a way that the amount of variation is less than α. So, after executing each change, the distance of the generated variation from the input process is calculated. If the distance is less than α, another round of changes can be applied as long as the distance between the generated variant and the input process stays less than α. The pseudo code of the proposed algorithm is showed in Figure 5.b.
The change functions are selected randomly based on a probability distribution function. In the proposed tool, the user can select different probability distribution functions such as Guassian, Uniform, Beta and Gamma functions. Based on the selected distribution function and the user-defined variation rate, the variation functions and the corresponding change operations in structure tree would be applied.

In Figure 5.a, a screenshot of the proposed tool is shown. As explained earlier, we have built our tool on top of the PLG toolset. We have added a new option to the last version of PLG2 for generating new variants (it is marked in red in Figure 5.a). The user needs to set the number of variants that is desired. The variation rate should be set between 1 to 100 percent (it is set by default to 30%). Also, the user can define the probability distribution function for generating variants randomly. The generated variants can be selected and viewed in the left pane of the tool. In the future, we intend to further develop our tool as a plug-in for the ProM framework.

4 Conclusion

In this paper, we introduced a toolset for generating collections of business process variants according to a user-defined variation rate. We defined and classified various factors that can lead to the generation of a variant of an input process model based on which change functions can be defined. Then, we proposed an algorithm based on the structure-tree representation of input process models for applying these change functions. These functions would be performed based on different probability distribution functions and with respect to a user defined variation rate. Our toolset is implemented in PLG and is accessible to researchers.

References

1. Larosa, M., Dumas ,M., Uba ,R,.and Dijkman ,R. M.: Business Process Variability Modeling: A Survey .ACM Transactions on Software Engineering Methodology 22,2,11, 2013.
2. Larosa, M., Dumas, M., Uba, R.,and Dijkman, R. M., Business Process Model Merging: An Approach to Business Process Consolidation. ACM Transactions on Software Engineering Methodology 22,2,11, 2013.

3. Pourmasoumi, A.; Kahani, M.; Bagheri, E. and Asadi, M. Mining Common Morphological Fragments from Process Event Logs. In Proceedings of the 2014 Conference of the Centre for Advanced Studies on Collaborative Research, 2014.
4. Valenca, G., Alves, C., Alves, V., and Niu, N. A Systematic Mapping Study on Business Process Variability. Int. Journal of Computer Science & Information Technology 5,1, 2013.
5. Reichert, C. Li, M., and Wombacher, A., "The MINADEPT Clustering Approach for Discovering Reference Process Models out of Process Variants," International Journal of Cooperative Information Systems, vol. 19, no. 3-4, pp. 159–203, 2010.
6. Li, C. and Reichert, M. and Wombacher, A.: Mining Business Process Variants: Challenges, Scenarios, Algorithms. Data & Knowledge Engineering, 70(5), pp. 409-434, Elsevier, 2011.
7. Burattin, A., and Sperduti, A., PLG: A Framework for the Generation of Business Process Models and Their Execution Logs. In Business Process Management Workshops, pages 214–219. Springer, 2010.
8. Buijs, J. C. A. M., van Dongen, B. F., van der Aalst, Wil M. P., "Discovering and Navigating a Collection of Process Models using Multiple Quality Dimensions", in Proceedings of the 9th International Workshop on Business Process Intelligence, 2013.
9. Weske, M., Business Process Management: Concepts, Languages, Architectures. Springer-Verlag, Berlin, 2007.
10. Medeiros, A. K. A. d., and G¨unther, C. W., Process mining: Using CPN tools to Create Test Logs for Mining Algorithms, in Proceedings of the Sixth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools, vol. 576. Aarhus, Denmark, 2005.
11. Stocker, T., and Accorsi, R., "Secsy: Security-aware Synthesis of Process Event Logs," in Proceedings of the 5th International Workshop on Enterprise Modelling and Information Systems Architectures, St. Gallen, Switzerland, 2013.
12. Shugurov, I., Alexey A. Mitsyuk. Generation of a Set of Event Logs with Noise, Proceedings of the 8th Spring/Summer Young Researchers' Colloquium on Software Engineering, M.: 2014. P. 88-95, 2014.
13. J. C. A. M. Buijs, Flexible Evolutionary Algorithms for Mining Structured Process Models, Ph.D thesis, 2014.
14. Li, C., Mining process model variants: challenges, techniques, examples. Phd thesis. University of Twente, The Netherlands, 2010.
15. Pawlik, M., Augsten, N.: RTED: A Robust Algorithm for the Tree Edit Distance. CoRR, abs/1201.0230, 2012.

Early Validation of Control Software for Automation Plants on the Example of a Seawater Desalination Plant

Veronika Brandstetter[1], Andreas Froese[2], Bastian Tenbergen[2],
Andreas Vogelsang[3], Jan Christoph Wehrstedt[1], Thorsten Weyer[2]

[1] Siemens AG, Corporate Technology, Germany
{veronika.brandstetter|janchristoph.wehrstedt}@siemens.com
[2] paluno – The Ruhr Institute for Software Technology, Univ. of Duisburg-Essen, Germany
{andreas.froese|bastian.tenbergen|thorsten.weyer}@paluno.uni-due.de
[3] Technische Universität München, Germany
vogelsan@in.tum.de

Abstract. In automation plants, the software that controls the behavior of a plant must adhere to strict process requirements arising from the technical processes and from the physical plant design. In current practice, the validation of the control software starts late in the engineering process – in many cases not before the plant is almost completely constructed, leading to increased efforts for correcting revealed defects. Based on an industrial example, we propose an approach that allows early validation of automation software against the plant processes and assumptions about the physical plant design through simulation.

Keywords: Automation Technology, Process Plants, Desalination Plant, Context Modeling, Simulation, Validation, Executable Requirements, Models.

1 Introduction

The development of plants (e.g. processing facilities in the chemical industry, production facilities in factories, or baggage routing facilities at airports) is a complex planning and engineering task. Typically, such plants are designed individually and the entire construction process from the first idea until commissioning takes years, involving many different disciplines like process engineering, physical plant design, mechanics, electronics, and software engineering [1].

The *automation software* of the plant controls its various technical devices (e.g. valves, containers, or conveyor belts). The overall goal of the automation software is to control the involved processes safely, reliably, with sufficient quality, and with the lowest demand on resources, such as time, energy, and material input [2]. Validating whether the automation software satisfies the process requirements of the entire plant is typically conducted at a late stage in the plant development when technical processes and the physical plant design are already defined. However, it is widely acknowledged, that the later the control software of the plant is validated, the higher the effort for correcting revealed defects is, leading to budget overruns and project delays.

This research was funded in part by the BMBF under grants 01IS12005A, 01IS12005C, and 01IS12005R.

In this paper, we propose an approach that fosters early validation of automation control software against the plant process based on specified requirements of the control software and assumptions about the physical plant design. Our approach aims at identifying defects in the requirements for the automation software (in the sense of incorrectly or incompletely specified requirements). The key idea of the approach is to use simulation during early stages of plant development to assess the impact of the specified requirements for the automation software on the plant process. The approach primarily aims at validating requirements for control software of plants. Yet, we expect that our approach can be used widely to validate application software for technical systems with complex technical and physical incarnations, where assumptions about the physical design can be made early in the development process.

2 Running Example

We illustrate our approach by means of a seawater desalination plant[1]. Desalination plants are used to remove salts from seawater to produce drinking water. In our simplified running example, seawater is collected through a subsurface intake system of four beach wells drilled into the seashore. From there the salt water is pumped through pipelines to the seawater tank, where it is stored and pretreated with chemicals before desalination can take place. An overview of a typical plant architecture and the technical architecture of one such beach wells is shown in Fig. 1. In the automation industry, technical architectures are documented using piping and instrumentation diagrams (P&ID). The left hand side of Fig. 1 shows the P&ID for one beach well, which is equipped with a pump and a discharge valve through which water is advanced to the seawater tank, a bypass control valve to adjust the flow rate into the tank, and a set of sensors. These beach well components are controlled by the beach well software to ensure that beach well actuators are controlled according to strict process requirements. In the remainder of this paper, we will focus on the beach well software to satisfy the process requirements shown in Table 1.

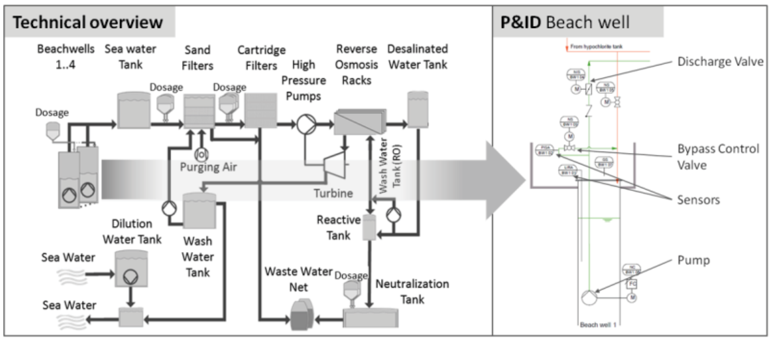


Fig. 1. Technical architecture of a typical seawater desalination plant and one beach well

[1] This running example is based on a free instructional DVD, see http://goo.gl/ppsNNo

Table 1. Process requirements for a beach well

ID	Process Requirement
Req 1	The pump must only run if filling level of the beach well tank is sufficient
Req 2	The pump load shall be minimized using the bypass control valve
Req 3	Discharge valve must be closed before pump starts
Req 4	Discharge valve must be open after pump has started
Req 5	Discharge valve must be closed after pump has stopped

3 Solution Approach

We seek to foster the early validation of automation software by means of simulation during early stages of the plant development process, i.e. when the plant is not yet finished, but when assumptions about the technical architecture of the plant can be made. The key idea of our approach (sketched in Fig. 2) is to document assumptions about the plant architecture explicitly by means of context models (see Section 3.1).

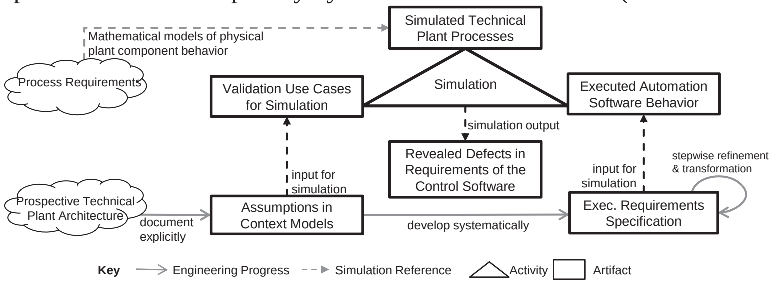

Simulated Technical Plant Processes
Mathematical models of physical plant component behavior
Process Requirements
Validation Use Cases for Simulation
Simulation
Executed Automation Software Behavior
simulation output
Revealed Defects in Requirements of the Control Software
stepwise refinement & transformation
input for simulation
input for simulation
Prospective Technical Plant Architecture
document explicitly
Assumptions in Context Models
develop systematically
Exec. Requirements Specification
Key
Engineering Progress
Simulation Reference
Activity
Artifact

Fig. 2. Overview over the approach

Context models often serve as a reference artifact in quality assurance approaches (see, e.g. [8]) and allow for systematically developing validation use cases (see Section 3.2) with execution semantics, which is necessary for the simulation of automation control software in particular (see, e.g., [9]). Based on the validation use cases, we derive an executable specification, configure the simulation tool, and execute the formal specification and check it against the validation use cases (see Section 3.3). The output of the simulation reveals incorrect or incomplete behavioral requirements of the automation software.

3.1 Assumptions about the Prospective Plant

We employ a number of diagrammatic representations, which we call operational context models, to document assumptions about the beach well's technical environment. These are introduced in the following.

Structural Operational Context Model (SOCM). The SOCM documents structural characteristics of the beach well's software, its physical components, and the components interacting with the beach well. This also entails dependencies between human users and automation software of other plant components, as shown in Fig. 3.

As can be seen, the SOCM depicts the interactions between the beach well components from Fig. 1 and documents the information exchange as well as relevant interfaces. Furthermore, interaction with human users is depicted as well as context influences (e.g., the beach wells pump water to a seawater tank).

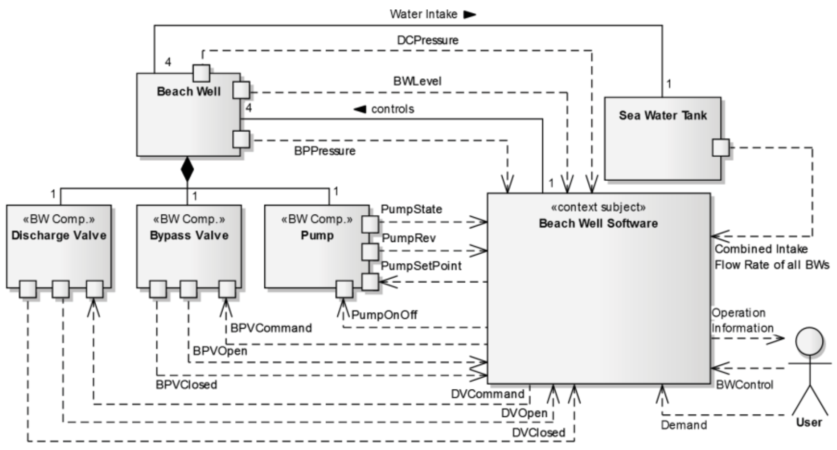


Fig. 3. Structural operational context model of the beach well control software

Functional Operational Context Model (FOCM). The FOCM documents the externally visible functions of the automation software and the physical plant. By abstracting from structural dependencies, the focus is on the functional dependency between the automation software and the physical plant components. This allows adopting a service-oriented view on the functionality by depicting only functions that influence each other. Fig. 4 depicts the FOCM of the beach well software and component functions. The software offers three externally visible functions by which the beach well component functions are controlled: "start beach well", "stop beach well", and "balance load". The signals from the SOCM in Fig. 3 were supplied with concrete values.

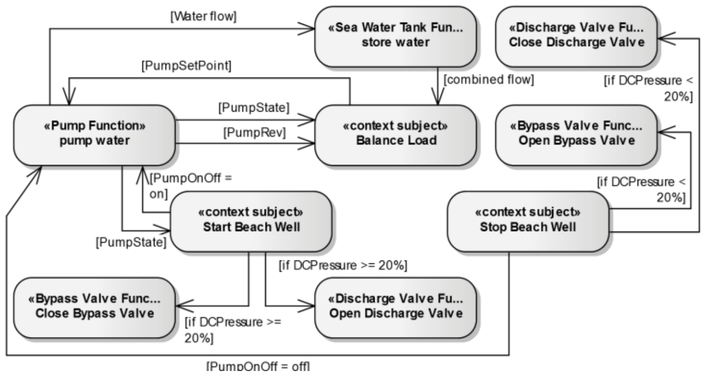

[Water flow]
«Sea Water Tank Fun... store water
«Discharge Valve Fu... Close Discharge Valve
[PumpSetPoint]
[combined flow]
[if DCPressure < 20%]
[PumpState]
[PumpRev]
«Pump Function» pump water
«context subject» Balance Load
«Bypass Valve Func... Open Bypass Valve
[PumpOnOff = on]
[if DCPressure < 20%]
«context subject» Start Beach Well
«context subject» Stop Beach Well
[PumpState]
[if DCPressure >= 20%]
«Bypass Valve Func... Close Bypass Valve
«Discharge Valve Fu... Open Discharge Valve
[if DCPressure >= 20%]
[PumpOnOff = off]

Fig. 4. Functional operational context model of the beach well control software

Behavioral Operational Context Model (BOCM). The BOCM documents the externally observable states of the physical plant components. Internal states of the plant components are abstracted and only states relevant to the automation software are documented. Transitions between states are triggered depending on the values of the signals from the FOCM. Fig. 5 shows an example of the BOCM of the beach well. As can be seen, the externally observable states of the beach well components from the SOCM are depicted as concurrent substates of the entire beach well. The guards on the transitions are conditions specified in the FOCM with regard to the beach well function "start beach well". Both bypass valve and discharge valve can be open or closed. The pump can either be off or on assume some intermediate states.

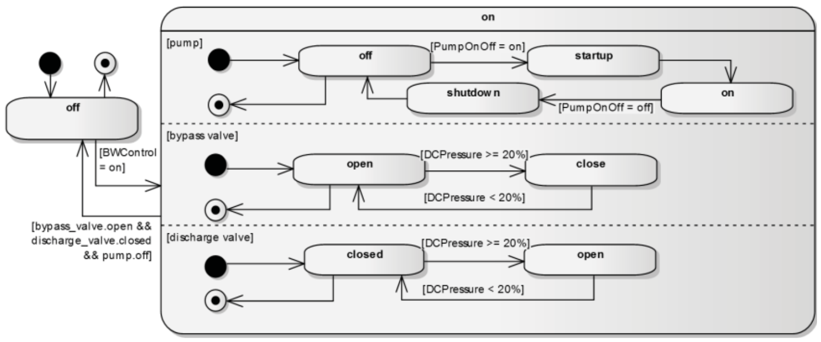


Fig. 5. Behavioral operational context model of the beach well and its components

3.2 Validation Use Cases

Based on the assumptions about the beach well's context (see Section 3.1), validation use cases are developed for every automation software function from the FOCM. Table 2 shows the validation use case for the FOCM function "Start Beach Well" using the Reqs 1, 3, and 4 from Table 1 as pre- and post-conditions regarding the startup procedure for the beach wells.

Table 2. Validation use case "Start Beach Well"

Title	Start Beach Well	
Description	A beach well is manually started by the user of the desalination plant	
Trigger	**BWControl == "The user initiates the start of the beach well".**	
Precondition	**bypass_valve.open == true && discharge_valve.closed == true**	
Postcondition	**pump.on == true && discharge_valve.open == true && bypass_valve.closed == true**	
Step	**Action**	**Actor**
1	The User initiates the start of a beach well	User
2	The Beach Well Software checks whether the beach well is in standby	Beach Well Software
3	The Beach Well Software closes the discharge valve	Beach Well Software
4	The Discharge Valve sends feedback that the valve is closed	Discharge Valve
5	The Beach Well Software starts the Pump with minimal revolutions	Beach Well Software
6	The Pump sends feedback that the Pump is started	Pump
7	The Beach Well Software closes the Bypass valve	Beach Well Software
8	The Beach Well Software opens the Discharge Valve	Beach Well Software
9	The Discharge Valve sends feedback that the valve is opened	Discharge Valve
10	The Beach Well Software reports that the beach well is on	Beach Well Software

Typically, use cases comprise a set of sequential steps called a scenario, which document interactions between the automation software and the physical plant components that lead to the desired post-conditions [4]. Since the process requirements demand a certain sequence of valve actuation and minimal tank filling level, the beach well software must perform several steps as documented in the scenario.

Simulation requires the scenario to be executable. Hence, we formalize the scenario using Message Sequence Charts (MSCs, [5]), as shown in Fig. 6. The MSC defines a sequence of messages based on the interface information and the plant signals from the SOCM (Fig. 3). These messages trigger transitions between states from the BOCM (Fig. 5). This formalized scenario serves as reference for the simulation of the beach well software and beach well technical process (see Section 3.3).

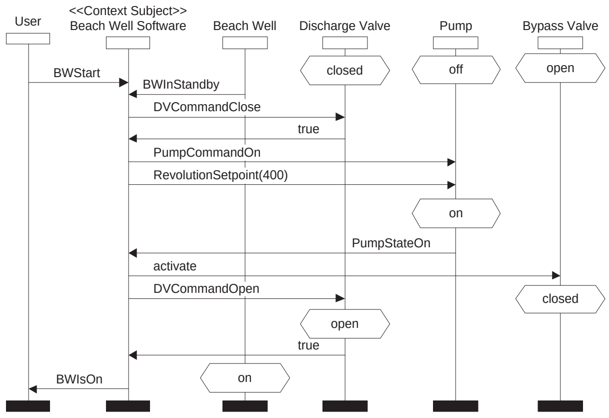


Fig. 6. Formalized scenario from the validation use case “Start Beach Well” in Table 2

3.3 Executable Requirements Specifications

After the validation use case was documented and the scenario was formalized, a simulation tool can be configured (e.g. in Aspen, MATLAB-SIMULINK, or Modelica). For this purpose, the simulated plant process (which gave rise to the process requirements from Table 1) must be modeled and coupled with the formalized scenario in a simulation process, which executes the validation use case.

Modeling the Simulated Plant Process. The plant process is described using algebraic equations representing the physical behavior of the beach well components. These equations can be taken from component libraries, in which the relevant configuration parameters (e.g. height of tank, length of pipes) are stored. Fig. 7 shows the steps involved in modeling the plant process of the beach well. The upper section of Fig. 7 depicts the relevant excerpt of the plant architecture taken from the SOCM (Fig. 3). The middle section of Fig. 7 shows the differential equations for flow, filling level, beach well tank pressure, and the seawater tank pressure. The process models of pump and discharge valve comprise two equations for flow and pressure. Since the four components are associated with three connecting pipes, six additional equations are necessary to describe the flow balance in the physical pipes and the pressure potential at the connection points. Once the relevant physical plant components and the equations representing their dynamic behavior are identified, the simulation tool can be configured (bottom section of Fig. 7)[2].

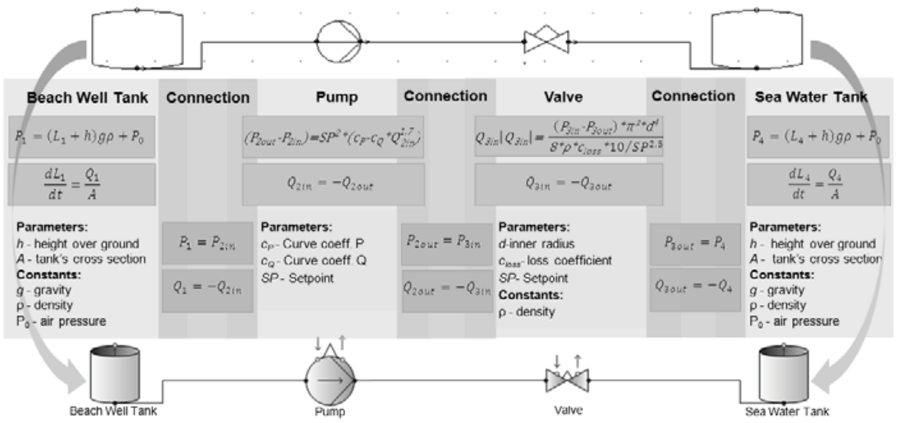


Fig. 7. Modeling the Simulated Beach Well Processes.

Coupling with the Executable Automation Software Behavior. The functional interplay between the beach well software and the plant process must be represented in the simulation tool. To this end, we structure the behavior of the automation software by functions [6], which subsume the process requirements from Table 1 and formalizes them by a behavioral model using the executable specification from Fig. 6.

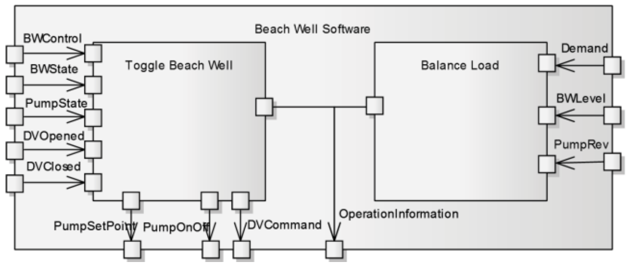


Fig. 8. SysML block definition diagram showing two beach well software functions

Fig. 8 shows two functions of the beach well software: “Toggle Beach Well” and “Balance Load”. Each function handles a subset of the input and output signals of the automation software specified in the SOCM (Fig. 3). The behavior of these functions must be specified by an executable behavior description (e.g. a state machine or a code snippet), which allows executing the scenario from Fig. 6, as shown in Fig. 9.

Simulation Process. Based on the formalized validation use cases (see Section 3.2), the equation system documenting the plant processes (Fig. 7) and the automation

[2] In this example, we have used the inhouse tool CoSMOS by Siemens, which can be used to configure and simulate automation plants in an object-oriented fashion, similar to Matlab.

software behavior (Fig. 8), simulation can be conducted by executing the beach well processes and emulating a physical process that takes place therein. The scenario from the validation use cases is used as input for the simulation. If the simulation tool can execute the scenario and the externally observable states from the BOCM match the final states of the physical plant components at the end of the simulation run, the requirements specification is valid with regard to that use case.

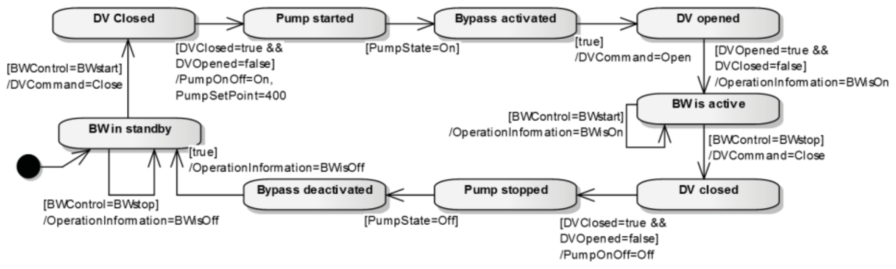


Fig. 9. Behavior of the “Toggle Beach Well” function described by a state machine

4 Conclusion

In this paper, we presented an approach, which enables the developers to validate the control software of automation plants early in the engineering process by first modeling assumptions about the design of the plant and subsequently deriving an executable specification which can be simulated. If no defects are revealed in the simulation it can be concluded that the behavior of the control software is valid with respect to the plant process and the assumptions about the plant design (assuming the behavior is implemented correctly). Albeit this approach was illustrated using an example of the automation industry, we believe it is applicable to any type of software system that controls real-world processes (e.g., business processes for information systems).

References

1. Löwen, U., Bertsch, R., Böhm, B., et a.: Systematization of the Engineering in Automation Plants. In: Automatisierungstechnische Praxis 4, pp. 54-61 (2005)
2. Wagner, T., Wehrstedt, J., Löwen, U., et al: Application and Evaluation in the Automation Domain. In: Model-based Engineering of Embedded Systems, Springer, (2012)
3. Davis, A.: Software Requirements: Objects, Functions, and States. Prentice-Hall (1993)
4. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley Professional (2001)
5. ITU-T Z.120: Formal description techniques – Message Sequence Chart, 02/2011.
6. Vogelsang, A., Eder, S., Hackenberg, G., et al.: Supporting concurrent de-velopment of requirements and architecture: A model-based approach. MODELSWARD 2014.
7. Broy, M.: Multifunctional software systems: Structured modeling and specification of functional requirements. In: Science of Comp. Prog. 75(12), pp. 1193-1214 (2010)
8. Jackson, M.: Problem frames. Addison-Wesley. Harlow (2006)
9. Li, D., Li, X., Liu, J., Liu, Z.: Validation of requirement models by automatic prototyping. Innovations in Systems and Softw. Eng. 4, pp. 241–248 (2008)

PRIVATEER: A Private Record Linkage Toolkit

Alexandros Karakasidis[1], Georgia Koloniari[2], and Vassilios S. Verykios[1]

[1] Hellenic Open University, School of Science & Technology, Patras, Greece
{a.karakasidis, verykios}@eap.gr
[2] University of Macedonia, Department of Applied Informatics, Thessaloniki, Greece
gkoloniari@uom.edu.gr

Abstract. Privacy preserving record linkage (PPRL) is the process of integrating data across multiple heterogeneous data sources without compromising their privacy. While many techniques have been developed for PPRL, there has not been, up to now, a universally accepted method providing both performance and quality of results in all cases. To this end, we present PRIVATEER, a toolkit which aims at enabling practitioners to compare various techniques involved in the PPRL process and determine the best for their needs. The toolkit is based on a simulator, designed to be highly configurable, modular and extensible, allowing the user to test different configurations by combining a number of privacy preserving blocking and matching methods with corresponding distance and similarity measures on her own or sample data. We showcase the usability of our toolkit by presenting experimental results measuring both quality and performance of state-of-the-art PPRL methods.

Keywords: Privacy, Record Linkage, Toolkit, Simulator

1 Introduction

Large volumes of data, describing individuals, are collected by public or private organizations. Being able to interconnect and analyze these independently stored data is beneficial for businesses, governments and research. For instance, being able to interconnect data of distinct health organizations, hospitals and physicians could prevent epidemic outbreaks or help us better understand the mechanisms behind certain diseases.

The task of linking such heterogeneous data, known as the *record linkage problem*, is not trivial [4]. Since data are stored by independent organizations, there is no unique identifier to determine common data. Also, such data suffer from low quality, exhibiting typos, abbreviations and misspellings, requiring approximate matching methods to link them. Additional challenges arise when we need to protect the privacy of the subjects described by the data. Therefore, *Privacy Preserving Record Linkage* (*PPRL*) has emerged as the problem where data from heterogeneous sources are integrated without revealing to the participating sources any extra knowledge not relating to their common records.

To privately link data with accuracy, *Privacy Preserving Matching* (*PPM*) methods focus on elaborately matching data, while maintaining privacy. Though accurate, matching methods often incur high computational costs. To increase their scalability, PPM methods are combined with *Privacy Preserving Blocking* (*PPB*) methods. PPB methods prune out unlikely to match candidate pairs of records, while also maintaining privacy. Even though they improve efficiency, PPB methods often compromise matching quality.

To the best of our knowledge, there is no single solution that successfully addresses the privacy preserving record linkage problem to its full extent. In particular, state-of-the-art PPB and PPM methods usually focus on a specific type of data. For instance, there are PPM approaches appropriate for string data [11, 16] and others for numerical data [9, 10]. Moreover, there are two party methods and others that require the deployment of a third party. As a result, each time that a PPRL solution has to be deployed, the most appropriate algorithms for the given application scenario have to be selected.

To this end, we introduce *PRIVATEER*, a PRIVATE REcord linkage toolkit designed to assist practitioners in this selection process, and also serve as a comparison tool for different methods. PRIVATEER is a modular simulator, constantly under development, implementing a set of state-of-the-art PPRL methods. Considering the various tasks involved in PPRL, the aim of PRIVATEER is to accommodate all this functionality into a single toolkit. It consists of clearly separate modules, each of them performing a separate task, and uses different components implementing various privacy preserving matching and blocking methods, and auxiliary components that are deployed by matching and blocking methods, such as different distance measures used in approximate matching and reference set generators, where a reference set is used as an intermediate point of comparison. Highly composable, PRIVATEER allows all compatible PPB methods to be combined with compatible PPM ones, and with corresponding auxiliary methods, so as to form a complete PPRL solution.

PRIVATEER is also configurable enabling the user to fine tune each of the deployed methods and provides a variety of performance measures. While the toolkit is equipped with sample data, it also enables the use of a user's own data. Towards extensibility, PRIVATEER's clear and comprehensive design provides an API for extending its functionality. Through the Java paradigm the user is able to add its own modules and algorithms.

Similar toolkits, such as TAILOR [6], have been developed for the classical record linkage problem. Related tools for PPRL include data generators [3], corrupters [2], and tools that combine both functionalities [18]. While these tools aim at developing test data for PPRL, PRIVATEER aims at testing different PPRL methods. A first version of our toolkit is presented in [14], but without offering composability or a graphical user interface.

The rest of this paper is organized as follows. In Section 2, we present the architectural design of PRIVATEER and briefly describe the PPM and PPB algorithms, distance measures, and reference set generation methods currently implemented in the toolkit. Section 3 provides experimental results derived from PRIVATEER and we conclude in Section 4.

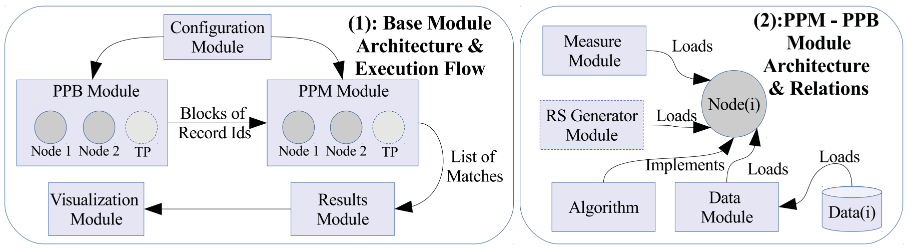


Fig. 1. Architecture of PRIVATEER

2 PRIVATEER's Architecture

The primary goals of PRIVATEER's design is composability and extensibility. We wanted PRIVATEER to easily enable the combination of different methods to compose a complete PPRL solution, and the incorporation of new ones as state-of-the-art PPM and PPB methods are constantly developed. To this end, an object-oriented design is appropriate, thus, we used Java to build the simulator comprising PRIVATEER. Also, Java has the property of *Write Once Run Anywhere*, providing machine independence, and there are many out-of-the-box libraries and data structures one can exploit.

We have made a particular effort to limit the assumptions when designing the architectural components of the simulator. In particular, we assume that all operations are executed sequentially. In many PPB and PPM algorithms, there are functions that require the parallel operation of the matching parties. PRIVATEER simulates this functionality by sequentially executing separate objects. Also, we only focus on the processing cost and do not examine communication costs and data propagation delays.

2.1 PRIVATEER's Modules

We followed a modular design, separating data, execution functionality and implemented methods. This approach improves PRIVATEER's extensibility, as it is very easy through its API to add new features and modules. Moreover, since data is retrieved in a transparent way, new data sources may be added as well.

Next, we present the architectural elements that comprise PRIVATEER.

The Base Module. The base module holds PRIVATEER's main functionality organizing the execution of each evaluation experiment. It coordinates the different modules and enables us to execute each experiment multiple times. As illustrated in Fig. 1(1), it hosts the configuration module, a PPB and a PPM module, the results processing and the visualization module.

The Configuration Module. The configuration module is invoked by the base module and is responsible for setting an experiment's execution parameters. It consists of a GUI implemented in JavaFX which receives input from the user and a mechanism that maps user input to specific parameters. This mapping mechanism is important, since each of PRIVATEER's modules has to be separately configured using separate parameters provided by the same user interface.

The Nodes. The nodes implement the functionality of the data sources and a third party, when one is needed. While a data source node has direct access to its data, the third party neither holds nor has direct access to any data. Instead, it is fed with data by the source nodes. Nodes are deployed by the PPB or PPM module and implement their part of the respective algorithm (Fig. 1(2)).

The Data Module. The data module is used to provide direct access to external data. It is invoked by each data source node. The used data may be stored in different databases and the configuration module enables the user to select her own input database, thanks to a transparent connectivity mechanism. Currently, MySQL and SQLite databases are supported.

The Measure Module. This module implements the distance or similarity measure to be used by the PPB and later PPM algorithm, in a transparent way. Thus, it allows us to deploy any compatible measure with any PPB and PPM method using the GUI. This way, the user has the ability to test the available algorithms and assess their behavior with different configurations.

The Reference Set Generator Module. Similarly to the measure module, the reference set generator provides different methods that can be used interchangeably with our PPB and PPM methods.

The PPB and PPM Modules. The privacy preserving blocking (PPB) and the privacy preserving matching (PPM) modules implement the corresponding type of protocols. Depending on the type of the protocol, as shown in Fig. 1(1), each module initiates two nodes and, when necessary, a third party node. All these nodes implement the same algorithm, which is chosen through the GUI and provided to the corresponding module by the base module. For a PPB or PPM algorithm to operate, the PPB or PPM module load a measure module, which implements the measure used by the corresponding algorithm. If it is required by the implemented algorithm, a reference set generator module is also loaded.

PPRL protocols may vary regarding their implementation and the number of participating nodes. Generally, however, they are either based on some type of data transformation or they may exhibit a multiparty computational behavior. In both cases, a third party may be required and all options may be implemented in PRIVATEER. Moreover, to offer the ability of experimenting with a no blocking setup, a mockup No Blocking method has also been implemented.

Regarding its operation, the PPB passes on the parameters received from the configuration module to the respective modules it hosts and executes the selected PPB algorithm. When the execution concludes, it produces lists of blocks of record ids which are then used by the PPM module. The PPM module receives the blocks of record ids from the PPB module and applies the selected PPM algorithm on each of them. Upon the conclusion of the execution of the PPM module, a list with matching record ids is produced which is then passed to the results processing module for processing.

The Results Processing Module. The results processing module processes the matches list received from the PPM module and calculates the number of total matches, the number of true and false matches, precision, recall and the execution times of the PPM and PPB modules. If the input data have the same unique identifiers (i.e., corrupted versions of the same dataset), the module automatically calculates the measurements based on the common primary key. Otherwise, a matching csv file with the matches between different primary keys is used.

The Visualization Module. The role of the visualization module is to visualize the results of each experiment by collecting the statistics produced by the results processing module and produce plots and tables of these statistics.

2.2 Implemented Algorithms and Measures

The modules of the architecture enable the selection of a method for each of the different tasks in the PPRL process. Thus, the PRIVATEER is equipped with a pool of implemented state-of-the-art approaches for each such task, i.e., PPB and PPM algorithms, distance and similarity measures and reference set generators (Tables 1-3).

The PPB algorithms currently implemented in PRIVATEER are illustrated in the upper part of Table 1. The first column holds the name of the algorithm, the second a short description, the third shows the type of data each algorithm is designed for and the forth indicates whether a third party node is required. Similarly, PPM algorithms are included in the lower part of Table 1.

Many of the PPRL algorithms that have been suggested, use a distance or similarity measure [4]. We designed PRIVATEER so as to offer the ability to use in the PPRL algorithms alternative measures, other than the default ones. Possible combinations are illustrated in the third column of Table 2. Default algorithms that use the measure are indicated by bold.

Reference sets (RS) are used in PPRL as an intermediate point of comparison to provide privacy. A reference set may be derived from a publicly available database or be arbitrarily generated. Table 3 summarizes PRIVATEER's currently available methods for RS generation. All methods apart from *Arbitrary String Generation* [16] use a user defined publicly available dataset.

3 Execution Results

To illustrate PRIVATEER's usability, we present sample experiments exploring alternative combinations of PPB and PPM methods, beyond their default setup, so as to assess their behavior and performance in different configurations.

All experiments are conducted using an Intel i3 PC with 8GB of RAM. We use a sample dataset, which originates from the North Carolina voters database[3]. We assume that the attributes: last name, first name, middle name, city of residence and precinct description comprise a candidate key, and use them for both matching and blocking. After deduplicating the dataset based on the selected candidate key, we generate two databases of 50 000 records each, so that the 25% of their records is common. Finally, we use the "master" table of Lahman's baseball database[4], as a source for reference set creation. Since we are interested

[3] Available at ftp://alt.ncsbe.gov/data/
[4] Available at http://www.seanlahman.com

	Name	Description	Datatype	3P
		PPB Algorithms		
B1	No Blocking	Option when the use of blocking is not desired. Records to be linked are organized in a single block for each of the two data sources.	N/A	
B2	SNEF [13]	SNEF is a meta-blocking algorithm specifically designed to shift the Sorted Neighborhood approach [8] to the PPRL paradigm. This algorithm uses reference set clustering for creating blocks.	Strings, Numbers	✔
B3	TPPB [19]	A single attribute is utilized for blocking. Each data source creates a separate reference set from a publicly available database. Then, it merges its data with the reference set and lexicographically sorts the result.	Strings	
B4	Simple Blocking [1]	Introduces hash signature transformations to securely represent TF/IDF weight vectors. Blocks consist of records which share common strings called tokens, with a token being an intact word. The Jaccard metric is used for comparing hash signatures.	Strings	✔
B5	Phonetic Blocking [12]	Uses Soundex's inherent privacy characteristics, which are based on the common information suppression property of all phonetic encoding algorithms. Records are organized in blocks based on their phonetic encoding similarity.	Strings	✔
		PPM Algorithms		
M1	Bloom Bigrams [17]	Bigrams of strings are inserted into a Bloom Filter. Then, a bitwise comparison is performed on Bloom Filters using the Dice Coefficient.	Strings	✔
M2	Embedding Spaces [16]	Based on the edit distances between actual and reference set data, a transformation of the data, called embedding, is derived and used for matching.	Strings	✔
M3	Secure Edit Distance [12]	Secures the use of edit distance through the use of Bloom filters. Each character of a matching field is extracted and after appending its position in the string, it is inserted in a Bloom filter which is encrypted using a secure hash function.	Strings	
M4	Homomorphic Encryption [9, 10]	Homomorphic encryption manages to calculate differences securely based on the properties of specific cryptographic algorithms, such as RSA.	Numbers	
M5	Soundex Matching [11]	Transforms attributes into Soundex codes and exploit its inherent information suppression properties to provide privacy.	Strings	✔
M6	Hash Signatures [1]	Calculates the TF/IDF weights of all words, called tokens, within all fields. Then, a hash function is used to create arrays of these weights, one for each record. This method is not suitable for approximate matching, since the entire token (word) is encoded.	Strings	✔

Table 1. Supported PPB and PPM algorithms.

Measure	Description	Compatibility
	Distance measures	
Levenshtein	Calculates the minimum number of operations (character insertion, deletion and substitution) required to transform one string to another.	**B2, B3, M2**
Damerau - Levenshtein	Alternative to Levenshtein distance that also takes into account the transposition of two adjacent characters.	B2, B3, M2
	Similarity measures	
Jaro - Winkler	Focused on short strings such as names. Produces a normalized score between 0, when there is no similarity, and 1, where there is an exact match between two strings.	B2, B3, M2
Jaccard coefficient	Used to measure set similarity. In the context of PRIVATEER, it is used with bit vector representations of strings, as in [1].	**B4**, M1
Dice's coefficient	Also used with bit vector representations of strings. Used as a default similarity measure in [17].	B4, **M1**
Q-grams Dice's coefficient	Dice's coefficient for plain text comparisons in combination with q-grams, as in [7].	B2, B3, M2

Table 2. Supported measures.

Method	Description	Compatibility
NN-Clustering	Tries to produce the most representative reference set from the entire data corpus.	M2, B2, B3
K-Medoids Clustering	Elects medoids from the most representative reference set from the entire data corpus.	M2, **B2**, B3
Arbitrary String Generation	Generates random strings	**M2**, B2, B3
Durstenfeld Shuffle	Generates a reference set of specific size. Given an array of elements the first n of them are randomly exchanged with others in the array and selected out of a data corpus.	M2, B2, B3

Table 3. Reference set generators.

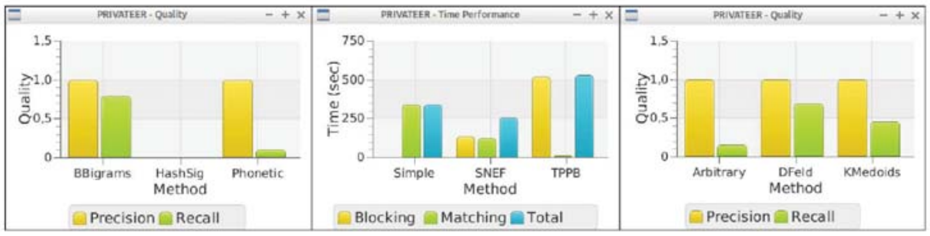


Fig. 2. Results illustrated by PRIVATEER's visualization module: (left) PPM, (center) PPB and (right) RS generation.

in linking low quality data, we corrupt one of the two databases using the corrupter described in [2]. The corrupted records contain one error per attribute so that a join with the original ones returns no matches. With equal probability, an error might either be a character insertion, deletion or substitution.

Figure 2(left) illustrates how PRIVATEER allows us to test a blocking method with different matching methods. In particular, we combine the TPPB method [19], with three PPM methods: Bloom Bigrams [17], originally used in [19] as well, Hash Signatures [1], and Phonetics Matching [11]. We compare the three combinations in terms of precision and recall. As we observe in Fig. 2(left), Hash Signatures have zero recall, since this method is not designed for approximate matching. On the other hand, the Bloom Bigrams' high performance in terms of recall justifies its overall popularity [10].

In the second experiment, we combine a matching method with different blocking methods. In particular, Bloom Bigrams is combined with Simple Blocking [1], SNEF [13] and TPPB [19]. As for blocking, efficiency is important, we measure time to determine the most efficient blocking method for Bloom Bigrams matching. Figure 2(center) illustrates the time required for PPB, for PPM and for the overall PPRL process. We observe how the matching time varies depending on the blocking method used. While Simple Blocking is the fastest blocking approach, it does not manage to significantly reduce the matching time. On the other hand, while SNEF is slower while blocking, it achieves the best overall time as it significantly reduces matching time. Finally, TPPB has the highest blocking time, but achieves the best matching time. It clearly creates the optimal blocks, but unfortunately the time blocking requires is greater than the overall time for SNEF.

Next, we evaluate three reference set generation methods with SNEF[13] and use Bloom Bigrams for matching [17]. The first one is the arbitrary string generation method [16], the second is a modification of the Durstenfeld shuffle algorithm [5], while the third uses the K-medoids algorithm [15], which was originally used in [13]. Figure 2(right) shows that with regards to precision and recall, the Durstenfeld Shuffle, and not the K-medoids used in the default setup of SNEF, is the best choice.

4 Conclusions

We have presented PRIVATEER, a modular, highly configurable privacy preserving record linkage toolkit that enables combining and comparing different PPRL solutions. We showed the usability of our toolkit by including evaluation results derived from experiments run by the PRIVATEER on sample data.

Acknowledgment. Attendance to this conference was partially supported by the Hellenic Artificial Intelligence Society (EETN).

References

1. Al-Lawati, A., Lee, D., McDaniel, P.: Blocking - aware private record linkage. In: IQIS (2005)
2. Bachteler, T., Reiher, J.: A test data generator for evaluating record linkage methods. Tech. rep., German RLC Work. Paper No. wp-grlc-2012-01 (2012)
3. Christen, P.: Febrl-: an open source data cleaning, deduplication and record linkage system with a graphical user interface. In: ACM SIGKDD (2008)
4. Christen, P.: Data Matching. Data-Centric Systems and Applications, Springer (2012)
5. Durstenfeld, R.: Algorithm 235: Random permutation. Communications of the ACM 7(7), 420– (Jul 1964)
6. Elfeky, M.G., Verykios, V.S., Elmagarmid, A.K.: Tailor: a record linkage toolbox. In: IEEE ICDE (2002)
7. Gravano, L., Ipeirotis, P.G., Jagadish, H.V., Koudas, N., Muthukrishnan, S., Pietarinen, L., Srivastava, D.: Using q-grams in a dbms for approximate string processing. IEEE Data Engineering Bulletin 24(4), 28–34 (2001)
8. Hernández, M.A., Stolfo, S.J.: Real-world data is dirty: Data cleansing and the merge/purge problem. Data Mining & Knowledge Discovery 2(1), 9–37 (1998)
9. Inan, A., Kantarcioglu, M., Bertino, E., Scannapieco, M.: A hybrid approach to private record linkage. In: IEEE ICDE (2008)
10. Inan, A., Kantarcioglu, M., Ghinita, G., Bertino, E.: Private record matching using differential privacy. In: ACM EDBT (2010)
11. Karakasidis, A., Verykios, V.S.: Privacy preserving record linkage using phonetic codes. In: BCI (2009)
12. Karakasidis, A., Verykios, V.S.: Secure blocking + secure matching = secure record linkage. JCSE 5(3), 223–235 (2011)
13. Karakasidis, A., Verykios, V.S.: A sorted neighborhood approach to multidimensional privacy preserving blocking. In: IEEE ICDMW (2012)
14. Karakasidis, A., Verykios, V.S.: A simulator for privacy preserving record linkage. In: EANN, Part II (2013)
15. Kaufman, L., Rousseeuw, P.: Clustering by Means of Medoids. Reports of the Faculty of Mathematics and Informatics. Delft Univ. of Technology (1987)
16. Scannapieco, M., Figotin, I., Bertino, E., Elmagarmid, A.K.: Privacy preserving schema and data matching. In: ACM SIGMOD (2007)
17. Schnell, R., Bachteler, T., Reiher, J.: Privacy preserving record linkage using bloom filters. BMC Medical Informatics & Decision Making 9(1), 41+ (2009)
18. Tran, K., Vatsalan, D., Christen, P.: Geco: an online personal data generator and corruptor. In: ACM CIKM (2013)
19. Vatsalan, D., Christen, P., Verykios, V.S.: Efficient two-party private blocking based on sorted nearest neighborhood clustering. In: ACM CIKM (2013)

STS-Tool 3.0: Maintaining Security in Socio-Technical Systems

Mattia Salnitri, Elda Paja, Mauro Poggianella, and Paolo Giorgini

University of Trento, Trento, Italy
{mattia.salnitri, elda.paja, poggianella, paolo.giorgini}@unitn.it

Abstract. In this paper, we present STS-Tool 3.0: a software tool that helps security requirement engineers in maintaining high level of security in socio-technical systems. STS-Tool 3.0 allows to specify social/organizational security requirements and to enforce them in part of the implementation of socio-technical systems.

1 Introduction

Socio-technical systems are an interplay of social (human and organizations) and technical subsystems, which interact with one another to reach their objectives. Examples of socio-technical systems are smart cities, air traffic management systems and payment systems, i.e. systems where banks, payment services and e-shops interact to transfer monetary values.

Subsystems in socio-technical systems are autonomous, heterogeneous and weakly controllable, but they highly depend on each other. For example, in the payment system the banks and payment services are autonomous but without interacting with each other, both of the systems will not be able to transfer monetary values.

In such interconnected environment is central maintaining a high level of security: a security issue of one subsystem may cause a chain reaction and impact many other subsystems. For example, in a payment socio-technical system, a security issue on the information integrity of the payment orders in one of the banks might compromise the entire system: all the banks and payment services that use the compromised bank may have received payment orders containing unauthorized modifications.

Therefore, it is essential maintaining the level of security specified by the stakeholders, i.e. being sure that security requirements are enforced in socio-technical systems' implementation.

In previous work [10], we have proposed a framework for specifying security requirements, checking their enforcement in business processes executed in such systems and generating a part of the implementation. Specifically, the framework permits to: (i) specify social-organizational security requirements; (ii) generate, from social-organizational models, business processes, i.e., specifications of how subsystems operate and interact; (iii) generate procedural security policies from social-organizational security requirements; (iv) verify security policies against business processes; (v) generate part of the implementation. We provide methodological guidance for each and every activity. The detailed process and phases are presented in [11].

In this paper we illustrate STS-Tool 3.0, the tool that supports the framework proposed in [10, 11]. Specifically, the paper describes how STS-Tool 3.0 supports each phase and the salient new features of STS-Tool 3.0.

The paper is structured as follows. Section 2 introduces the architecture and the implementation of STS-Tool 3.0, while Section 3 describes the content of the demonstration. Section 4 contains the related work and Section 5 concludes.

2 STS-Tool 3.0: architecture and implementation

STS-Tool 3.0 has the ambitious objective of supporting the generation of the implementation of socio-technical systems that enforces of social/organizational security requirements.

Such security requirements are based on high-level of abstraction concepts, such as goals/objectives of actors and delegation of goals/objectives, but, due to the conceptual gap between these concepts and the implementation, it is not possible to directly generate the code that enforces the security requirements.

STS-Tool 3.0 uses business process models to bridge the conceptual gap. It generates business processes, from social/organizational models, that are refined by security requirement engineers and then it transforms such business processes in the implementation. Specifically it uses using STS-ml [4, 7], a goal-based modeling language, for social/organizational models, and SecBPMN2-ml [9], a modeling language for business processes and security, for business process models.

The generation of a secure implementation of socio-technical systems needs business processes that enforce the social/organizational security requirements. STS-Tool 3.0 verifies the enforcement of security requirements by transforming them in security policies, i.e., security requirements in terms of business process elements, and it verifies such policies against the business processes.

STS-Tool 3.0 extends version 2.0, with a number of features, the most relevant are specified below:

- generation of business processes from the social/organizational perspective;
- generation of procedural security policies from social/organizational security requirements;
- verification of business processes against procedural security policies;
- generation of part of implementation of socio-technical system.

Figure 1 shows the architecture of STS-Tool 3.0. The components are divided in three parts: *Model editors* allow the visualization and the modification of models; *Model transformers* are used to transform models; *Automated reasoners* are used to check the consistency of the models and check the enforcement of business process against security policies.

STS-ml editor and *STS-ml compliance verifier* components are part of STS-Tool 2.0. They facilitate the creation of STS-ml models and the automated verification of their coherency. The components added in STS-Tool 3.0, highlighted with a thick border in Fig. 1, are described below.

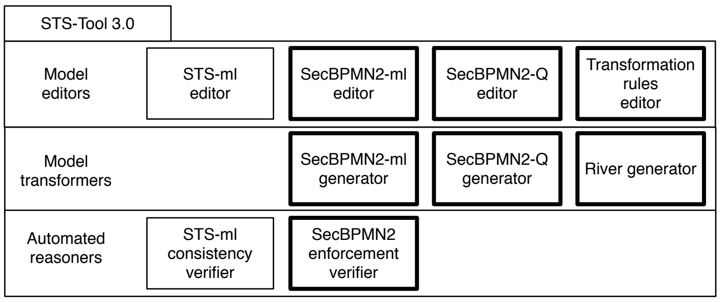


Fig. 1. Components of STS-Tool 3.0.

SecBPMN2-ml editor It permits to specify and to modify business processes with security aspects, using SecBPMN2-ml. This components extends BPMN2 Modeler[1]: an Eclipse[2] plug in for drawing BPMN 2.0 diagrams

SecBPMN2-Q editor It permits to model security policies using SecBPMN2-Q. This component, like *SecBPMN2-ml editor*, extends BPMN2 Modeler.

SecBPMN2-ml generator It automatically generates SecBPMN2-ml diagrams from STS-ml diagrams. SecBPMN2-ml diagrams are generated already filled with SecBPMN2-ml elements that can be derived from STS-ml diagram.

SecBPMN2-Q generator It generates SecBPMN2-Q security policies from security requirements specified in STS-ml diagram.

Transformation rules editor This component permits to visualize and modify transformation rules, i.e. set of rules used to generate the SecBPMN2-ml and SecBPMN2-Q models from, respectively, STS-ml models and the list of social/organizational security requirements.

SecBPMN2 enforcement verifier It is a verification engine that verifies if SecBPMN2-Q security policies are enforced in SecBPMN2-ml business processes. It generates a list of SecBPMN2-ml elements that do not satisfy the security policy and a list of STS-ml elements that are connected to the business process that does not satisfy the security policy. These elements are highlighted in the editors to facilitate their identification.

River generator It generates part of socio-technical system implementations. The component uses River [14], a scripting language for specifying business artifacts and the business logic associated to such artifacts. The generated code implements the management of the data used in the business processes. STS-Tool 3.0 does not generate the functional part of the implementation, i.e. the part of the implementation that executes the actions modelled by the activities of the business processes. The information contained in the strings which identify each activity, is not enough to generate the implementation that execute the activity. For further details, see [8].

[1] https://www.eclipse.org/bpmn2-modeler/
[2] https://www.eclipse.org

2.1 SecBPMN2 enforcement verifier component

The *SecBPMN2 enforcement verifier*, described in this subsection, is the core part of STS-Tool 3.0. We used $DLV^{\mathcal{K}}$ [5] for verifying SecBPMN2-ml business processes against SecBPMN2-Q security policies. $DLV^{\mathcal{K}}$ is a planner: a software engine, based on DLV [6], that generates plans, i.e., sequences of actions, that achieve a set of goals and do not violate given constraints. We transformed business processes in constraints and security policies in goals. Therefore, if $DLV^{\mathcal{K}}$ generates a plan, the security policies are satisfied against the business processes. For further details see [12].

Figure 2 shows the architecture of the component: grey boxes are software components, white boxes are input/output artifacts, while dashed arrows connect the inputs/outputs of the component to internal components.

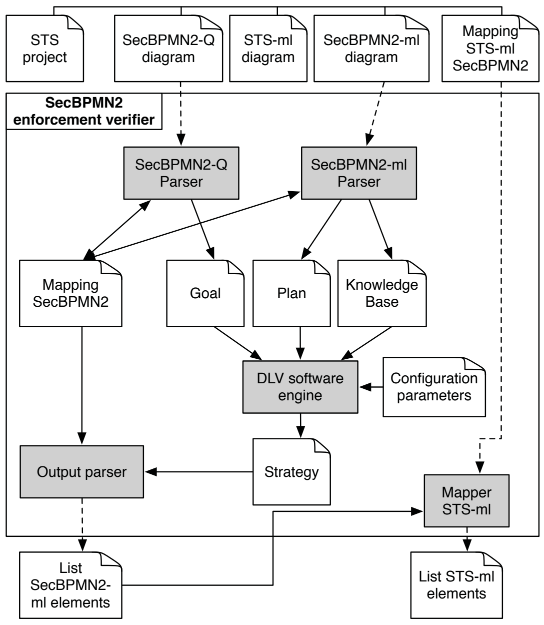


Fig. 2. Architecture of SecBPMN2 enforcement verifier.

SecBPMN2 enforcement verifier receives in input *STS project* that is composed of: *STS-ml diagram*, *SecBPMN2-ml diagram*, *SecBPMN2-Q diagram* and *Mapping STS-ml-SecBPMN2*. The latter part of the input contains the links between STS-ml elements and SecBPMN2-ml elements. It is created during the generation of SecBPMN2-ml diagrams from STS-ml diagrams. *SecBPMN2-ml parser* receives in input a *SecBPMN2-ml diagram* and it creates a *Plan* and the *Knowledge base*, i.e., the list of elements of the business process. Similarly *SecBPMN2-Q parser* component receives in input a *SecBPMN2-Q diagrams* and it creates a set of *Goals*. The *DLV software engine* has strict limitations on the names of the variables and constants, therefore, the *SecBPMN2-ml parser* and *SecBPMN2-Q parser* generate names that are compliant with the DLV limitations and store in a file called *Map IDs SecBPMN2* the relations between the name of activities, in SecBPMN2-ml diagrams, and the generated names. The *DLV software engine* checks the *Goal* against the *Plan* using the *Knowledge based* and it generates, if possible, a plan, i.e. a list of actions. *Output parser* uses the *MapIDs SecBPMN2* to generate, from the plan, a sequence of SecBPMN2-ml elements that, if executed, will satisfy the security policies. Finally, *Mapper STS-ml* uses the list and *Mapping STS-ml-SecBPMN2* to generate the list of *STS-ml elements*.

3 Demonstration content

We illustrate STS-Tool 3.0 with the payment engine example (Example 1) following the phases of the process shown in Fig. 3.

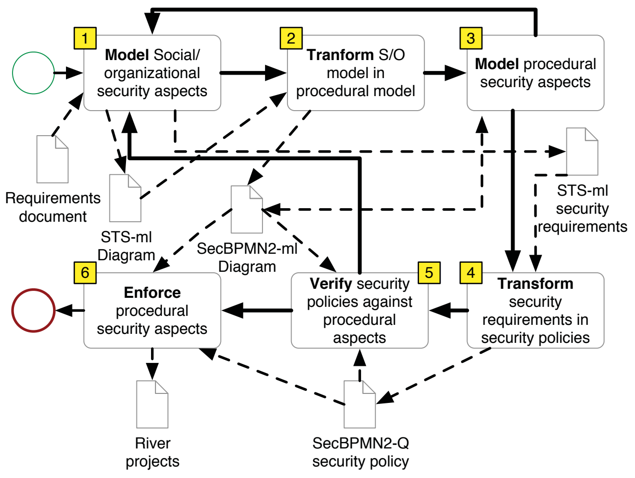

1
Model Social/ organizational security aspects
2
Tranform S/O model in procedural model
3
Model procedural security aspects
Requirements document
STS-ml Diagram
SecBPMN2-ml Diagram
STS-ml security requirements
6
Enforce procedural security aspects
5
Verify security policies against procedural aspects
4
Transform security requirements in security policies
River projects
SecBPMN2-Q security policy

Fig. 3. Methodological process from [11].

Example 1 (Payment Engine). SAP Payment Engine (PE) [13] is a software system created to perform payments for e-commerce shops or, more in general, for services that allow users to pay with electronic methods. Usually, to support the electronic payments, the e-shops implement interfaces to communicate with all the banks they intend to use as source/destination of the payments. But each bank requires a different set of protocols and security measures. Therefore the e-shops are forced to put a noticeable amount of effort to implement different interfaces, and for medium/small companies it is not acceptable investing large quantity of time and money just to allow people to pay the goods/services they offer. The PE minimizes such effort: it contains a set of interfaces with the most known banks in the word that can be used out of the shelf. E-shop *programmers only have to create one interface to transmit the required data to the PE system. The payment system is a socio-technical system since it involves customers, banks, the PE, etc., which interact with each other to perform payments.* □

Phase 1 It supports the modeling of socio/organizational aspects of the socio-technical system under consideration. Figure 4 shows a screen shot of STS-Tool 3.0 with an STS-ml diagram in the upper part. Such modelling language focuses on representing the main stakeholders (e.g. *PE* and *Bank* in Fig. 4), their objectives, as well as their interactions (e.g. Fig. 4 the objective *Transfer performed* is delegated from the *PE* to *Bank*). Most importantly, it captures security requirements. For example, in Fig. 4 *Auth* (authorization) and *Ava* (availability) security requirements are specified.
Phase 2 It deals with the transformation of social/organizational models to business processes, receiving in input a social/organizational diagram and generating a procedural diagram of the system-to-be. STS-Tool 3.0 uses SecBPMN2-ml as modelling language for business process with security aspects. It permits to model participants (e.g., *PE* and *Bank* in SecBPMN2-ml diagram shown in Fig. 4) that execute activities (e.g., *Perform transfer* in Fig. 4) and exchange messages (e.g. *Transfer order* in Fig. 4). Such modeling language permits to specify security concepts using security annotations, such as the orange solid circle in Fig. 4 that represents confidentiality of the message transmitted in the communication between the two participants.
Phase 3 Social/organizational models do not contain the information required to generate complete SecBPMN2-ml models. Therefore, this phase is executed to enrich the SecBPMN2-ml models generated by phase 2 with details such as the temporal aspects and security choices on the executed processes.
Phases 1-3 are repeated until security requirement engineers decide they have captured all important (security) details and the procedural model is complete and accurate.
Phase 4 It generates procedural security policies from social/organizational security requirements. This phase permits to translate security requirements, defined in terms of social/organizational concepts, in more operational constraints, as the ones specified in security policies. STS-Tool 3.0 uses SecBPMN2-Q as modeling language for security policies. Figure 4 shows an example of SecBPMN2-Q diagram. Such modelling language extends SecBPMN2-ml, since it permits to use relations such as path, represented with a dashed arrow in Fig. 4.
Phase 5 It deals with the verification of procedural security policies, generated in phase 4, against the procedural model, enriched in phase 3. This phase validates SecBPMN2-ml models by verifying if they enforce the security requirements specified in the social/organizational model. If all security policies are enforced, then the security requirement engineers have the proof that the procedural model meets the security requirements. If the procedural model does not satisfy all procedural security policies, the process starts from the beginning.
Phase 6 It consists in generating part of the implementation from the enriched and verified procedural model. The resulting application will enforce the social/organizational security requirements, because social/organizational security requirements define the security policies that are verified against the procedural model. Therefore, because the transformation enforces all the security aspects defined in the procedural model, the implementation can be considered secure. For further details, see [8].

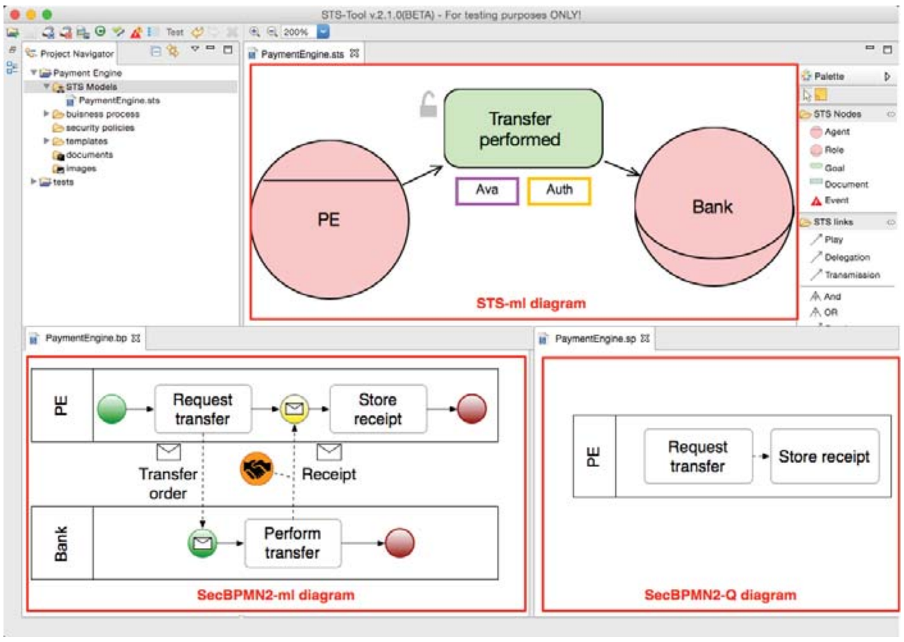


Fig. 4. Screenshot of STS-Tool 3.0.

4 Related work

A number of software tools, e.g., [17, 1, 2], can be used by security requirement engineers for specifying or for designing part of secure socio-technical systems. For example Secure Tropos [17] and STS-Tool 2.0 [16] are focused on specifying security requirements at social/organizational level. While other software tools, such as Adonis [1] or SNP-BPA [15] are focused on the analysis of business processes. For what concerns the implementation of business processes, Altova [3] and Alfresco [2] permit to define and execute business processes executed both by technical components and humans. But they do not generate part of the implementation: they, instead, call a service for each activity that is not executed by humans.

As far as our knowledge goes, no software tool assists security requirement engineers in enforcing security requirements from their early specification until the implementation in socio-technical systems.

5 Conclusion and future work

STS-Tool 3.0 is ready for public use. This version of the tool is the result of an iterative development process, having been tested on multiple case studies and evaluated with practitioners [11]. It has proven suitable to model and reason over models of a large size from different domains [11, 12], such as Air Traffic Management Control and Telecommunications. Future work about STS-Tool 3.0 includes: (i) integration of other languages for the generation of the implementation of socio-technical systems; (ii) implementing a plug in management system that allows for adding functionalities to STS-Tool 3.0.

Acknowledgement

This research was partially supported by the ERC advanced grant 267856, 'Lucretius: Foundations for Software Evolution', `www.lucretius.eu`.

References

1. Adonis Website. Last visited May '15. http://www.boc-group.com/it/products/adonis/.
2. Alfresco Website. Last visited May '15. http://www.alfresco.com.
3. Altova Website. Last visited May '15. http://www.altova.com.
4. F. Dalpiaz, E. Paja, and P. Giorgini. Security Requirements Engineering via Commitments. In *STAST'11*, pages 1–8.
5. T. Eiter, W. Faber, N. Leone, G. Pfeifer, and A. Polleres. Planning under incomplete knowledge. In *CL '00*, pages 807–821.
6. N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV system for knowledge representation and reasoning. *ACM Trans. Comput. Logic*, 7(3):499–562, July 2006.
7. E. Paja, F. Dalpiaz, and P. Giorgini. Managing Security Requirements Conflicts in Socio-Technical Systems. In *ER '13*, pages 270–283.
8. M. Salnitri, A. Brucker, and P. Giorgini. From Secure Business Process Models to Secure Artifact-Centric Specifications.
9. M. Salnitri, F. Dalpiaz, and P. Giorgini. Modeling and Verifying Security Policies in Business Processes. *BPMDS '14*, pages 200–214.
10. M. Salnitri, E. Paja, and P. Giorgini. Preserving compliance with security requirements in socio-technical systems. In *Proc. of CSP*, pages 49–61, 2014.
11. M. Salnitri, E. Paja, and P. Giorgini. From socio-technical requirements to technical security design: an sts-based framework. Technical report, DISI - University of Trento, 2015.
12. M. Salnitri, E. Paja, and P. Giorgini. Maintaining secure business processes in light of socio-technical systems' evolution. Technical report, DISI - University of Trento, 2015.
13. SAP Payment Engine Website. Last visited March '15. www.sap.com/ services-support/svc/custom-app-development/cnsltg/prebuilt/payment-engine/index.html.
14. SAP SE. *SAP River Developer Guide*, 2014. Document Version 1.0 – 2014-08-21, SAP HANA SPS 08, revision 82.
15. SNP-BPA Website. Last visited May '15. http://www.snp-bpa.com.
16. STS-Tool Website. Last visited May '15. http://www.sts-tool.eu.
17. Tropos Website. Last visited May '15. http://www.troposproject.org.

Extending Software Development Methodologies to Support Trustworthiness-by-Design

Nazila Gol Mohammadi[1], Torsten Bandyszak[1], Sachar Paulus[2], Per Håkon Meland[3], Thorsten Weyer[1] and Klaus Pohl[1]

[1]paluno – The Ruhr Institute for Software Technology, University of Duisburg-Essen, 45127 Essen, Germany
{nazila.golmohammadi, torsten.bandyszak, thorsten.weyer, klaus.pohl}@paluno.uni-due.de
[2]Mannheim University of Applied Sciences, Paul-Wittsack-Straße 10, 68163 Mannheim, Germany
s.paulus@hs-mannheim.de
[3]SINTEF ICT, Strindveien 4, N-7465 Trondheim, Norway
per.h.meland@sintef.no

Abstract. People are increasingly concerned about the trustworthiness of software that they use when acting within socio-technical systems. Ideally, software development projects have to address trustworthiness requirements from the very early stages of development using constructive methods to enable trustworthiness-by-design. We analyze the development methodologies with respect to their capabilities for supporting the development of trustworthy software. Our analysis reveals that well-established development methodologies do not specifically support the realization of trustworthy software. Based on findings, we propose a generic mechanism for extending development methodologies by incorporating process chunks that represent best practices and explicitly address the systematical design of trustworthy software. We demonstrate the application of our approach by extending a design methodology to foster the development of trustworthy software for socio-technical systems.

Keywords: Trustworthiness, Trustworthiness-by-design, Software Development Methodology

1 Introduction

Trustworthiness is a major issue for the development of software-intensive socio-technical systems [1, 2]. For instance, for the users of today's web applications and services it becomes increasingly difficult to track or control who stores personal and business-critical data. Thus, software-intensive systems need to be trustworthy to address concerns of their users and thereby foster the trust in these systems. Understanding how to address trustworthiness in early design phases is crucial for the successful development of software systems. Software development methodologies and processes should address the different challenges of engineering trustworthy software.

Trustworthiness is an important quality that needs to be engineered. There is a strong dependency between the degree of trustworthiness an information system exhibits and the suitability of the applied development methodology [3].

There are limited contributions that approach the trustworthiness issues other than those related to security. Most existing approaches assume that one-dimensional properties of services lead to trustworthiness of such services, and even to trust in it by users, such as a certification (e.g., Common Criteria [4]), the presence of certain technologies (e.g., encryption), or the use of certain methodologies (e.g., SSE-CMM [5]). In contrast, the Trusted Software Methodology [3] as a comprehensive and holistic methodology that explicitly focuses on trustworthiness is not flexible, since it is based on a certain development process. Though in principle the application of any development process model may result in trustworthy products, commonly used and well-established methodologies, such as user-centered [6] or test-driven development [7], do not specifically foster the systematic establishment of trustworthiness properties within the system. In order to address this gap, we believe that specific techniques and developer guidance should be defined as generic and reusable process building blocks. Defining reusable process chunks that can be integrated into well-established development methodologies instead of defining yet another development methodology brings flexibility, enables a powerful process modeling tool support, and reduces the complexity of tailoring already established development processes.

First, we review and analyze well-established software development methodologies by studying their characteristics that are promising to build trustworthy information systems, and the ones indicating improvement potential. In this paper, we build upon an outline of our approach sketched in [8], and provide a generic mechanism for enhancing software development by incorporating process chunks that explicitly address and enable trustworthiness-by-design. In particular, we propose an extension of the Software Process Engineering Meta-model (SPEM) [9], which allows for integrating and tailoring certain “trustworthy” process chunks into different development methodologies. These capability patterns can represent a broad range of trustworthiness-related practices, such as the preparation for run-time maintenance [10]. As an example, we analyzed the User-Centered Design (UCD) methodology [6] with respect to trustworthiness potentials and drawbacks as an example. Based on these findings, we demonstrate our approach by exemplarily extending the UCD process model.

The remainder of this paper is structured as follows: Section 2 provides a brief overview on the fundamental notions of trust and trustworthiness of STS. Section 3 presents our approach for extending development methodologies by trustworthiness-by-design capabilities and illustrates its application by showing how a popular engineering methodology can be extended to support trustworthiness-by-design. Section 4 summarizes the paper and gives an outlook on future work.

2 Fundamentals and Related work

Trust is defined as “a bet about the future contingent actions of others” [11]. Regarding software-intensive socio-technical systems (STS), which include humans, organizations, and the information systems [7], the scope of this definition can be broadened in order to include these systems as potential trustees. Because of delegation of tasks to STS, it can be said that the trustworthiness of such systems is a key concern that needs to be fostered and even engineered into these systems to maintain high levels of trust within society. Trustworthiness requirements are project-specific, and depend on domain and application. Software trustworthiness is highly dependent on the prescribed, yet evolving, set of requirements, technical decisions, and management decisions throughout the development process life cycle. A comprehensive list of trustworthiness attributes (e.g., correctness, reliability, safety, usability, security) should be taken into consideration when developing trustworthy software [2]. Hence, we focus on a multitude of software quality attributes that contribute to trustworthiness as analyzed in [2]. For example, trustworthiness may be evaluated with respect to the availability, confidentiality and integrity of stored information, the response time, or accuracy of outputs [12, 13, 14].

To the best of our knowledge, the Trusted Software Methodology (TSM) [3] is the only comprehensive approach that describes processes and guidance for engineering and assessing trustworthy software. It covers multiple quality attributes, and focuses on processes instead of evaluating development artifacts. TSM provides a set of Trust Principles, which describe established development practices or process characteristics that enhance software trustworthiness. A development process can be assessed by means of five different levels of trustworthiness, according to the conformance to the trust principles. This also constitutes the basis for process improvement with respect to trustworthiness. Though the principles constitute general best practices, the methodology, however, is assumed to be applied following a military standard for software development [15]. In contrast, our focus is on enhancing a broad spectrum of general software development methodologies in order to incorporate the consideration of trustworthiness and use them to create trustworthy software.

Yang et al. [3] review a set of software development methodologies in order to derive a meta-model for trustworthy development processes. They define process trustworthiness as "the degree of confidence that the software process produces expected trustworthy work products that satisfy their requirements" [3]. The meta-model includes, for example, trustworthy products that depend on a trustworthy process. It also depicts the connection to trustworthiness requirements. For modeling process trustworthiness, they adopt the Process Area concept from CMMI [16] and extend it by the Trust Principles, then constituting Trustworthy Process Areas (TPAs) [3]. The TPAs, in turn, can be refined by three categories, i.e. regarding trustworthiness assurance, trustworthiness monitoring, and trustworthiness engineering process areas. Thus, the approach covers the whole system life-cycle. Yang et al. also present their efforts towards designing a comprehensive Trustworthy Process Management Framework, which e.g., additionally involves a measurement model based on metrics [3].

In contrast, our approach relies on the SPEM [7], which provides a meta-model for describing software development processes. In our approach, we will use the Delivery Process and Capability Pattern concepts from SPEM. Capability patterns are process building blocks that are independent of specific process phases, and represent best development practices to be incorporated into a process [7]. The Delivery Process and Capability Pattern concepts originate from the SPEM. SPEM provides adequate concepts that allow for describing capability patterns on a fine-granular level, i.e. assigning concrete tasks, responsible roles, guidance, or involved artifacts.

The concepts introduced can be compared to the work of Yang et al. [3]. However, we propose a different structure and different concepts, e.g., using SPEM capability patterns instead of CMMI process areas (cf. [16]), or combining design and assessment in one meta-model.

3 Integrating Trustworthiness-by-Design in Development Methodologies

Characteristics of Trustworthiness-by-Design Processes. In order to incorporate the notion of trustworthiness-by-design into development methodologies, we consider and extend the SPEM meta-model [7] by specializing the *Delivery Process* concept so that it subsumes trustworthiness-by-design processes. We also utilize the concept of *Capability Patterns*. We define a *Trustworthy Product* (i.e. work product, development artifact) as a product that holds a range of its trustworthiness attributes for satisfying its trustworthiness requirements. Fig. 1 shows a corresponding ontology for Trustworthiness-by-Design Processes. The meta-model presented here shows the concepts that we have introduced in addition to SPEM (highlighted in grey in Fig. 1), specifically: *Trustworthiness-by-design Process* is a specialization of a delivery process and contains a set of capability patterns. A properly applied trustworthiness-by-design process will create a *Trustworthy Product* that exhibits certain trustworthiness attributes to meet its *Trustworthiness Requirements*. Trustworthiness requirements specify requirements that a Trustworthy Product should fulfill. *Assessment Model* verifies if the trustworthiness requirements have been met. Metrics could be used to evaluate the products. *Trustworthiness Evidence* is some kind of evidence to show that a trustworthiness-by-design process has been followed. Though this will not guarantee trustworthiness, it is at least an indication that planned measures have been taken into account to ensure it.
We define capability patterns that particularly address trustworthiness to improve existing design process models. For describing capability patterns, we provide the necessary content, e.g., concrete tasks, responsible roles, guidance, and involved artifacts [7].

An exemplary capability pattern for trustworthiness-by-design is the identification of threats and mitigating controls. This capability pattern involves analyzing system models or specifications in order to anticipate risks that might corrupt the system's trustworthiness across the whole life-cycle (e.g., also considering system operation).
To provide tool support for designing, tailoring and sharing trustworthy development processes, we use the Eclipse Process Framework (EPF[1]), which has an underlying

[1] Eclipse Process Framework Project (EPF), `http://www.eclipse.org/epf/`

meta-model based on SPEM. The EPF is a customizable software process-engineering framework for authoring, tailoring, and deploying development processes. All our capability patterns are organized in a plug-in that can be imported into any EPF project, which again can be exported to online process handbooks.

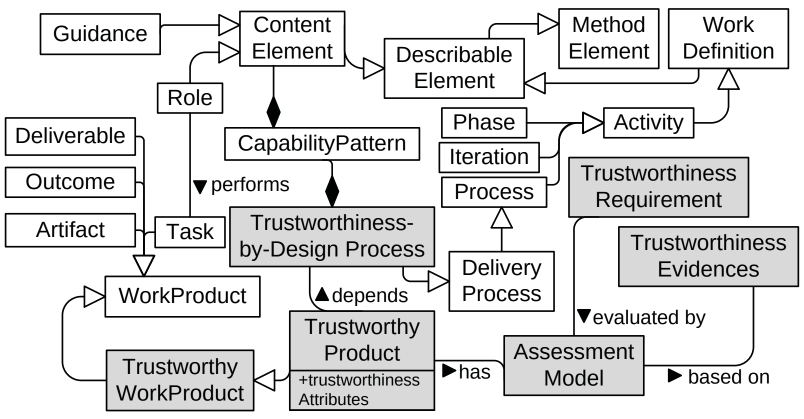


Fig. 1. Ontology for Trustworthiness-by-Design Processes

Extending the User-centered Design Methodology. The nature of engineering of trustworthy systems is different from simply engineering usable software. The key here is that trustworthiness is a subjective value judgment of stakeholders in a STS. There is a need to understand what trustworthiness attributes of the system will enhance the trust of a stakeholder in that system and how system design can thus help to circumvent any distrust-related concerns that the stakeholders have about the service. This makes it necessary to not only elicit requirements with respect to the way in which people will use the system, as would be done in a standard UCD [6] process, but also to draw up a set of requirements about which trustworthiness attributes will address the potential trust issues that the end users of the system highlight.

In order to assess the product with respect to the satisfaction of trustworthiness requirements the overall structure of the UCD approach can remain the same, with the only difference that in the process, besides usability and usefulness, trust and trustworthiness needs are specifically addressed.

We suggest the following extensions of the four major phases of the UCD methodology:

- In the initial *specify context of use* phase (Phase 1 in Fig. 2), a usability expert elicits from the future end users what the potential trust concerns are that they have with respect to using the system.
- In the specify user requirements phase (Phase 2 in Fig. 2), these concerns can then be turned into use case descriptions of situations in which the trust issues become apparent to the user. To this end, the Trustworthiness Capability Pattern “Identification of threats and mitigation controls” should be incorporated into UCD. By means of the involved analysis tools, threats to trustworthiness can be derived. It

should also be determined which controls can be applied in the design to mitigate the identified trustworthiness and trust issues.

- The produce design solutions phase (Phase 3 in Fig. 2) should then implement (e.g., in a prototype) the identified trustworthiness requirements.
- The "Measurement of end-to-end trustworthiness" capability pattern can enhance the *evaluation against requirements* phase by providing appropriate metrics and measurement approaches to validate that the system satisfied the required trustworthiness level (this can enhance Phase 4 in Fig. 2).

Name: User-Centered Design Process
Description: User-centered design processes [6] consist of the following general phases:

1) Knowledge elicitation and attempt at understanding the context of use;
2) Defining user requirements;
3) Prototyping the system and
4) Evaluation, which provides input for the refinement of the design.

This process model is generally used iteratively and by going through the process multiple times, developers converge on a user-friendly system.

Elements interesting for trustworthiness:

- User-centered design is a specialization of incremental development and therefore shares the same trustworthiness characteristics.
- By using an incremental user-centered process, it is possible that throughout the design process the design is validated to establish whether the trustworthiness attributes designed into the system appropriately address any concerns with respect to trust that the system users might have.

Improvement potential:

- Documenting trustworthiness requirements and thereafter generation of trustworthiness evaluation results for explicit documentation of trustworthiness evidences in order to support designers when making design decisions. Additionally, these documents bring awareness about the designed system to the end-users.
- Involvement of end-user to derive their trustworthiness expectations and to evaluate the system design towards the satisfaction of those expectations.

Usability for modeling trustworthiness: The user centered design processes are unrelated to trustworthiness modeling. Only the use of modeling techniques in general for a user centered design will enable to also model trustworthiness requirements.

Fig. 2. Indicative trustworthiness analysis for User-Centered Design

Fig. 3 illustrates the extension of UCD by plugging one of the proposed capability patterns namely "Identify Threats and Controls" in early stage of specifying user requirements.

As we have already shown above, the extension of the UCD process can be supported by using the EPF Composer. Fig. 4 shows an excerpt from the description of the extended UCD methodology as part of a software development process model (i.e. delivery process as defined by SPEM).

As the excerpt sketches, the corresponding "*Trustworthy User Centered Design*" methodology integrates the two additional capability patterns "Identification of threats and mitigation controls" (Fig. 4, Index no. 6) and "Measurement of end-to-end trustworthiness" (Fig. 4, Index no. 13) into specific phases of the original user-centered design process model.

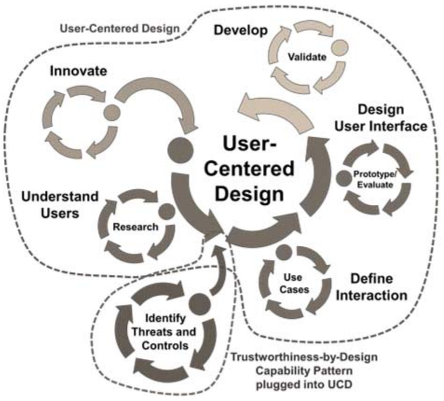
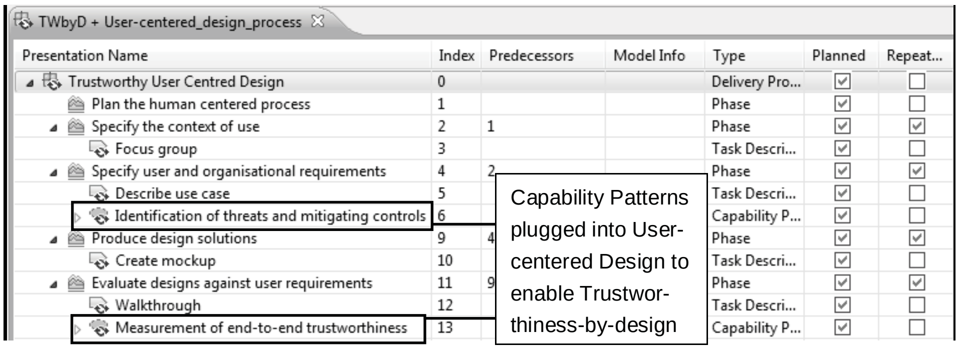


Fig. 3. Extending the User-Centered Design[2] for enabling Trustworthiness-by-Design [8]



TWbyD + User-centered_design_process

Presentation Name	Index	Predecessors	Model Info	Type	Planned	Repeat...
Trustworthy User Centred Design	0			Delivery Pro...	✓	☐
Plan the human centered process	1			Phase	✓	☐
Specify the context of use	2	1		Phase	✓	✓
Focus group	3			Task Descri...	✓	☐
Specify user and organisational requirements	4	2		Phase	✓	✓
Describe use case	5			Task Descri...	✓	☐
Identification of threats and mitigating controls	6			Capability P...	✓	☐
Produce design solutions	9	4		Phase	✓	✓
Create mockup	10			Task Descri...	✓	☐
Evaluate designs against user requirements	11	9		Phase	✓	✓
Walkthrough	12			Task Descri...	✓	☐
Measurement of end-to-end trustworthiness	13			Capability P...	✓	☐

Capability Patterns plugged into User-centered Design to enable Trustwor-thiness-by-design

Fig. 4. Enhancing the UCD methodology with trustworthiness capability patterns

4 Conclusion and Future Work

Existing software design methodologies have some capacities in ensuring security and a few other trustworthiness attributes. However, the treatment of a complete set trustworthiness attributes and requirements in software development is not yet well studied. We analyzed development methodologies for trustworthy development.

As a result, we concluded that none of them fully assures or addresses the development of trustworthy software. Consequently, individual activities, so-called "trustworthy development practices", must be identified and tailored into these processes in order to proceed towards systematically developing trustworthy software. The concept and an initial set of reusable, trustworthiness-enhancing process chunks in the form of Capability Patterns have been introduced. We have observed that the usage of appropriate trustworthiness capability patterns increases the confidence that the software development processes will result in trustworthy software.

Our work is still in progress, and the main ideas and findings will be further investigated. Further work is needed to evaluate the recommended extensions to these

[2] Based on http://www.sapdesignguild.org/editions/edition10/ucd_overview.asp

methodologies, how to combine capability patterns and investigate how trustworthiness attributes can be treated in a measurable and comparable way.

References

1. Whitworth, B.: A Brief Introduction to Socio-technical Systems. In: Encyclopedia of Information Science and Technology, pp. 394–400. IGI Global (2009)
2. Gol Mohammadi, N.; Paulus, S.; Bishr, M.; Metzger, A.; Könnecke, H.; Hartenstein, S.; Weyer, T.; Pohl, K.: Trustworthiness Attributes and Metrics for Engineering Trusted Internet-based Software Systems. In: Cloud Computing and Service Science 2013 (Selected Papers from CLOSER), CCIS, Springer (2013)
3. Yang, Y., Wang, Q., Li, M.: Process Trustworthiness as a Capability Indicator for Measuring and Improving Software Trustworthiness. In: Trustworthy Software Development Processes, Int'l. Conf. on Software Process. LNCS, vol. 5543, pp. 389-401. Springer (2009)
4. International Organization for Standardization: ISO 15408-1, Common Criteria, Information technology -- Security techniques -- Evaluation criteria for IT security. International Standard (2009)
5. International Organization for Standardization: ISO/IEC 21827, Information technology, Security techniques, Systems Security Engineering -- Capability Maturity Model® (SSE-CMM®). International Standard (2008)
6. Sutcliffe, A. G.: Convergence or Competition between Software Engineering and Human Computer Interaction. In: Human-Centered Software Eng. - Integrating Usability in the Software Development Lifecycle, Human-Computer Inter. Series, vol. 8, pp. 71-84 (2005)
7. Sommerville, I., Software Engineering. 9th Edition, Pearson, Boston (2011)
8. Gol Mohammadi, N.; Bandyszak, T.; Paulus, S.; Håkon Meland, P.; Weyer, T.; Pohl, K.: Extending Development Methodologies with Trustworthiness-By-Design for Socio-Technical Systems, In: Proceedings 7th Int'l. Conf. TRUST (2014)
9. Object Management Group: Software & Systems Process Engineering Meta-Model Specification, Version 2.0. Technical Report, Object Management Group (2008)
10. Bandyszak, T.; Gol Mohammadi, N.; Bishr, B.; Goldsteen, A.; Moffie, M.; Nasser, B. I.; Hartenstein, S.; Meichanetzoglou, S.: Cyber-Physical Systems Design for Runtime Trustworthiness Maintenance Supported by Tools. In: 1st Int'l. Workshop on Requirements Engineering for Self-Adaptive systems and Cyber Physical Systems (2015)
11. Sztompka, P.: Trust: A Sociological Theory. Cambridge University Press (1999)
12. Avizienis, A., Laprie, J. C., Randell, B., Landwehr, C.: Basic Concepts and Taxonomy of Dependable and Secure Computing. In: IEEE Transactions on Dependable and Secure Computing, vol. 1 issue 1, 11–33 (2004)
13. Gómez, M., Carbó, J., Benac-Earle, C.: An Anticipatory Trust Model for Open Distributed Systems. In: Anticipatory Behavior in Adaptive Learning Systems. LNCS, vol. 4520, pp. 307–324. Springer (2007)
14. Yolum, P., and Singh, M. P.: Engineering Self-Organizing Referral Networks for Trustworthy Service Selection. In: IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans vol. 35 no. 3, 396–407 (2005)
15. U.S. Department of Defense: Trusted Software Methodology, SDI-SD-91-000007, Volumes 1 and 2. Technical Report, U.S. Department of Defense, Strategic Defense Initiative Organization (1992)
16. Software Engineering Institute: Capability Maturity Model® Integration for Software Engineering, Version 1.1. Technical Report, Software Engineering Institute, Carnegie Mellon University (2002)

A Framework for Occupational Fraud Detection by Social Network Analysis

Sanni Lookman, Selmin Nurcan

Centre de Recherche en Informatique, Université Paris I, Panthéon-Sorbonne
lookman.sanni@malix.univ-paris1.fr, nurcan@univ-paris1.fr

Abstract. This paper explores issues related to occupational fraud detection. We observe over the past years, a broad use of network research across social and physical sciences including but not limited to social sharing and filtering, recommendation systems, marketing and customer intelligence, counter intelligence and law enforcement. However, the rate of social network analysis adoption in organizations by control professionals or even by academics for insider fraud detection purpose is still very low. This paper introduces the OFD – Occupational Fraud Detection framework, based on formal social network analysis and semantic reasoning principles by taking a design science research perspective.

Keywords: Design science, ontology, data mining, fraud detection, social network analysis, internal control, governance, risk, compliance.

1 Introduction

Frauds partly draw from human beings imaginative nature. Over the years, fraudster's attack methodologies have evolved from an opportunistic approach to some more sophisticated and traceless deception schemes and that, in a constantly yet automatizing but complexifying business environment. In recent years, several unethical behaviors within organizations have received significant attention. Celebrated cases range from financial scandals (Pechiney - 1988, Elf - 1994, Enron - 2001, Kerviel 2008) to data theft (WINDOWS 8 Beta - 2012, Korea Credit Bureau - 2014, SONY - 2014) and have proven that fraud is likely to happen at any level of an organization. The Association of Certified Fraud Examiners, in his 2014 report to the nations on occupational fraud and abuse [ACFE, 2014], estimates a global loss of 5% of revenues to fraud (3.7 trillion dollars if applied to the 2013 Gross World Product). They additionally reported that fraud cases were mostly uncovered by tips or chance (40%). That is an anonymous fraud hotline would even anticipate a lot of fraud damage and yet, knowledge discovery and data mining techniques are teeming.

Detection innovations include automated rules, watch lists matching, supervised and unsupervised classification, data fusion and link analysis. Such techniques have received increased industry specific interests for external frauds (i.e. committed by people outside of the organization) detection. Those would include cybercrimes by computer or network intrusion, credit card, insurance, telecommunication and credit application frauds [Phua et al., 2004], [Yufeng et al., 2004], [Cox et al., 1997], [Wheeler et al., 2000]. In the meantime, internal or occupational fraud, defined by the ACFE as the use of one's occupation for personal enrichment through the deliberate misuse or misapplication of the employing organization's resources or assets, has proved to be more prevalent than external fraud. PriceWaterhouseCoopers' 2014 Global Economic Crime Survey reports in France an average of 56% of internal fraud [PWC, 2014].

This paper elicits problems faced by investigators in the process of occupational fraud detection and comes up with a solution which contributes to solving these problems. Following the design science research paradigm [Wieringa, 2009], formal social network analysis and semantic modeling concepts have been reused to suggest a new perspective on the architecture of an effective fraud detection system.

The remainder of this paper is organized as follows. Section 2 introduces properties, formal analysis of social networks and motivation for their use to address fraud detection issues. Section 3 then demonstrates the design of the OFD framework and the validation of its design within a context of fraud detection from journal entries. In the last section, related works, concluding remarks and their implication for further research on social network analysis were taken up.

2 Social Networks

2.1 What is Social Network Analysis (SNA)?

A social network is a concept referring to a structure made of social actors sharing interests, activities, etc… Joseph Moreno is cited by most research papers on the topic of social network analysis as being the first to introduce methods and tools for a formal analysis. In the 1930s, he was the first one to use all four properties that characterizes SNA at the same time in a study aiming at explaining a spate of girls' runaways: (1) the intuition that links among social actors are important. (2) It is based on data that record social relations that link actors. (3)It draws heavily on graphic imagery to reveal and display the patterning of those links. (4) It develops mathematical and computational models to describe and explain those patterns [Freeman, 2011]. Basically, SNA aims at understanding relationships between the network participants, by means of mapping and measuring. SNA has received increased attention from organizations seeking to understand connection between patterns of interactions. It applies to a wide range of business problems including collaboration in workplaces, team building in post merger configuration, employee's engagement measurement, online reputation, customer intelligence, business strategy, disease contagion, counter terrorism, etc. It was a SNA which led US military to the capture of Saddam Hussein in December 2003 [PSU, 2007]. The tool Inflow for example is credited with contribution to the analysis of terrorist networks surrounding the September 11th events and contact tracing for HIV transmission in a state prison [INFLOW, 2010].

2.2 Formal Social Network Analysis

Whether used for infectious disease spread modeling, professional relations analysis, concentration of resources or power identification, SNA would follow two different approaches. Researchers distinguish between egocentric and socio-centric analysis of networks [Chung et al., 2006]. In the former type of analysis, the focus is made on local structure of networks, i.e. the network around a given node while the latter considers the network as a whole, looking at interactions patterns and the overall network structure by quantifying relationships between people. This distinction would impact the SNA process during data collection and graph visualization.

SNA provides both a visual and a mathematical analysis of relationships between the entities participating to the network. From the visual perspective, social networks are represented as “sociogram” [Scott et al., 2011] or graphs showing actors as nodes that are tied by one or many types of interdependency (values, ideas, visions, sex, friendship, kinship, collaboration, trade, antagonism, etc…). From a mathematical perspective, the social relations datasets translate into a matrix, underlining the visualized graph. This perspective serves at uncovering the graph’s theoretic properties (e.g.: number of edges, number of vertices, degree, multiplexity, centrality, density, closeness, betweenness, etc…), supported by metrics computed from the matrix, that help characterizing and even querying the network at hand.

2.3 Why Social Network Analysis for Addressing Insider Fraud Detection?

Social Network Analysis can bring value to occupational fraud detection in at least three ways. First, nowadays organizations are networked (staff, management, customers, suppliers, etc…) and fraud can originate from any part of the network. SNA brings the ability to analyze behaviors and reveal hidden connections that would have not been seen in raw text format.

Secondly, the dynamic nature of fraud makes detection challenging for the traditional rule based algorithms. Fraudsters are constantly adapting to circumvent the existing controls and any new pattern would not be covered by such static algorithms. As people excel at detecting patterns and their judgment when reviewing anomalous activities or transactions very valuable, we believe combining this human ability to computer’s capability to iteratively and tirelessly search for defined instances would improve the overall detection process.

Thirdly, SNA can help saving time during manual investigation, which is a necessary step for validating any potential fraud case uncovered by a tool. Traditional computer-aided audit tools are transaction oriented [ACL, 1987], output rows of incriminated transactions without the view of other related transactions performed by the same entities and thus make the manual investigation process labor intensive. With SNA the involved entities and their overall activities is readily available in a graph view for fraud examiners, who in turn are able to quickly visualize false positives and can focus on more risky cases.

3 The Occupational Fraud Detection Framework - OFD

3.1 Framework design

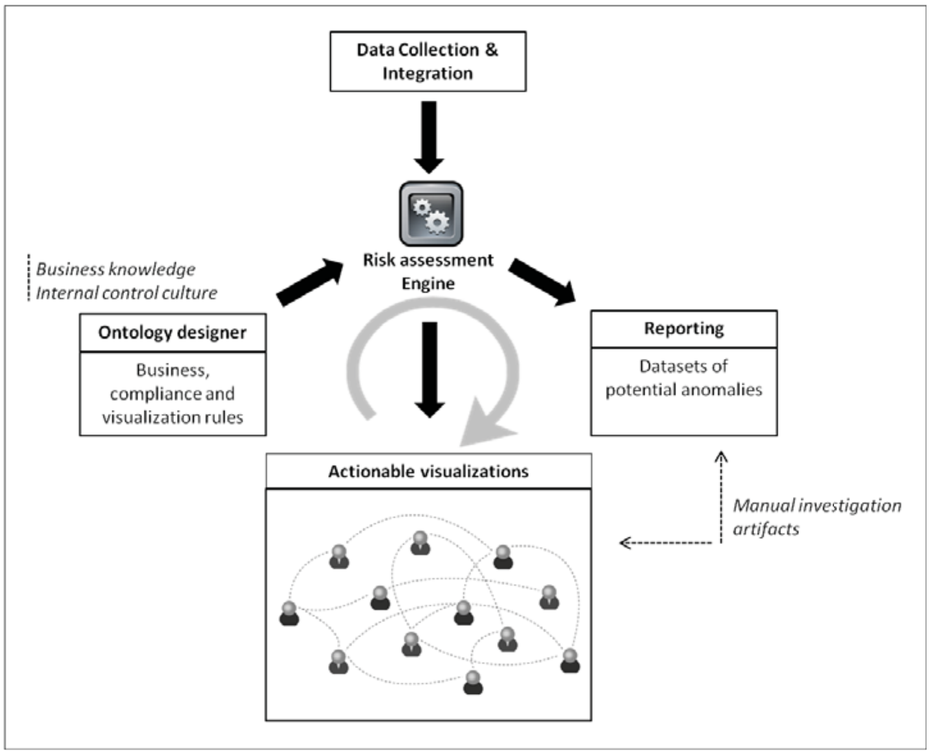

Data Collection & Integration
Risk assessment Engine
Business knowledge
Internal control culture
Ontology designer
Business, compliance and visualization rules
Reporting
Datasets of potential anomalies
Actionable visualizations
Manual investigation artifacts

Fig. 1. Overview of the OFD framework

The OFD framework as envisioned in this paper, starts with the selection of a process well integrated with IT, from which historical facts (e.g.: sales, journal entries, purchases, etc…) can be extracted from. OFD is built around four main components:

- The ontology designer, which is the specification component. As fraud examiners are barely proficient in data sciences to maintain robust systems their own and fraud schemes always evolving, the need of a layer of semantic in fraud detection systems architecture is mission critical. Actor types, interaction types and their respective characteristics are to be specified via this component. The way in which they are represented in the final sociogram (shape, color, etc…) is essential and to be defined here as well. Compliance related rules, conflicting interactions, antecedence or association rules are also regarded. The idea behind this component is its complete flexibility so as to enable a fraud examiner to not only control the rationale behind how fraudulent cases are uncovered, but also the display of the different visualizations available.

- The risk assessment engine is made of a data parser, for ensuring proper integration of raw data collected and a semantic reasoning system. The reasoner is meant to infer logical consequences from the rules specified in the ontology designer. It would analyze the parsed data with both socio-centric and ego-centric perspectives. At one hand, ego-centric analysis will highlight individual interactions which violate the set of rules specified, while at the other hand, socio-centric analysis will enable the identification of internal control deficiencies (e.g.: no segregation of duties) and the detection of fraud not pertaining to a specific transaction, or entity (e.g.: conflicts of interest, management frauds, etc…).
- The reporting component, like what exist today in the industry, would report on cases of violation of the specified rules, by outputting rows of potentially fraudulent transactions.
- The visualization component with its set of actionable sociograms includes a multidimensional social network view, showing several interaction types in the same network, what goes along with the socio-centric perspective mentioned earlier. Drill down and rollup capabilities would help zooming into transactions pertaining to a specific interaction type, or a specific actor of the network (ego-centric analysis). On reduced set of interactions, the time dimension would also be viewable. This component is critical to the overall detection process as through it, fraud examiners would uncover new unforeseen patterns to be specified in the ontology editor, thus paving the way for a continuously improving fraud detection engine.

3.2 OFD framework evaluation by early prototyping

Before jumping to the development of a generic, sound and theoretically grounded tool for supporting the framework introduced earlier, a review of the design has been performed. The aim of such evaluation was threefold:

a. Assess the extent to which graphs can fairly and faithfully represent the diversity of interactions happening between actors of the same or different organizations.
b. Measure the expressiveness of a social network in terms of red flagging of fraudulent interactions or transactions.
c. Gain insights on the perceived complexity by fraud examiners in the use of such visualizations to support fraud detection.

To this end, we ran a case study using accounting journal entries extractions as input for two different organizations of different size. The case study was conducted in collaboration with a population of internal auditors, who have been surveyed on various multidimensional social networks generated from the accounting journal entries (actionable visualization component). The number of auditors involved in the evaluation cannot be revealed in the presence of non disclosure agreement with the cooperating organization. The R project and the network analysis package “IGRAPH 0.7.1” [Csárdi et al., 2006] were used for scripting raw data parsing, business and design logic. Figure 2 illustrates the overall multidimensional network for one of the entities studied.

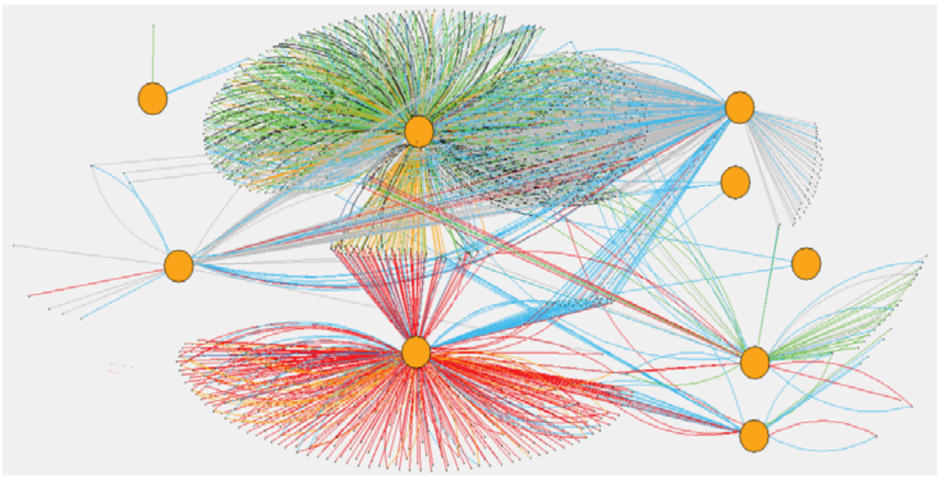


Fig. 1. Global multidimensional social network of accounting journal entries

Each edge in the graph above corresponds to a type of interaction happening between an employee (orange nodes) and a third party (other nodes - customer or supplier in this case). Red edges correspond to outgoing payments, orange ones being purchase invoices, etc… Different drills down or subsets of what is shown in figure 2 have been submitted to the auditors, like the one in figure 3, illustrating supplier only related interactions for the same entity as above.

The key takeaways from this evaluation exercise are as follows:

- Not all journal entries involve a third party (customer, supplier, etc…), what could be perceived as a threat to the validity of our social network oriented approach. Fortunately, such entries (depreciation, amortization, miscellaneous incomes, etc…) are usually subjected to rules which can be reasoned by the risk assessment engine and atypical entries solely highlighted in the reporting engine.
- The manipulation of the proposed visualizations is not that intuitive for auditors, even with a help document attached. Training should not be neglected as 20% of the surveyed auditors perceived the visualization as being too complex, embedding too much information at once. They actually did not provide any further answer to the questionnaire.
- The remaining participants’ high level observations or socio-centric conclusions were identical (e.g. non effectiveness of segregation of duties), denoting the good expressiveness of graphs for serving such purpose.
- At the other hand, the ego centric findings were diverse and varied from an auditor to another one, but not contradictory. The variability in the red flags of interest might be explained by the difference in the past experiences of each one of the au-

ditors. They tend to focus their testing procedures on the types of anomalies they expect to come across (what is quite aligned with traditional rule based static detection algorithms); the visualization can help then going beyond that, by expanding the range of possibilities and suggesting further investigation axes.

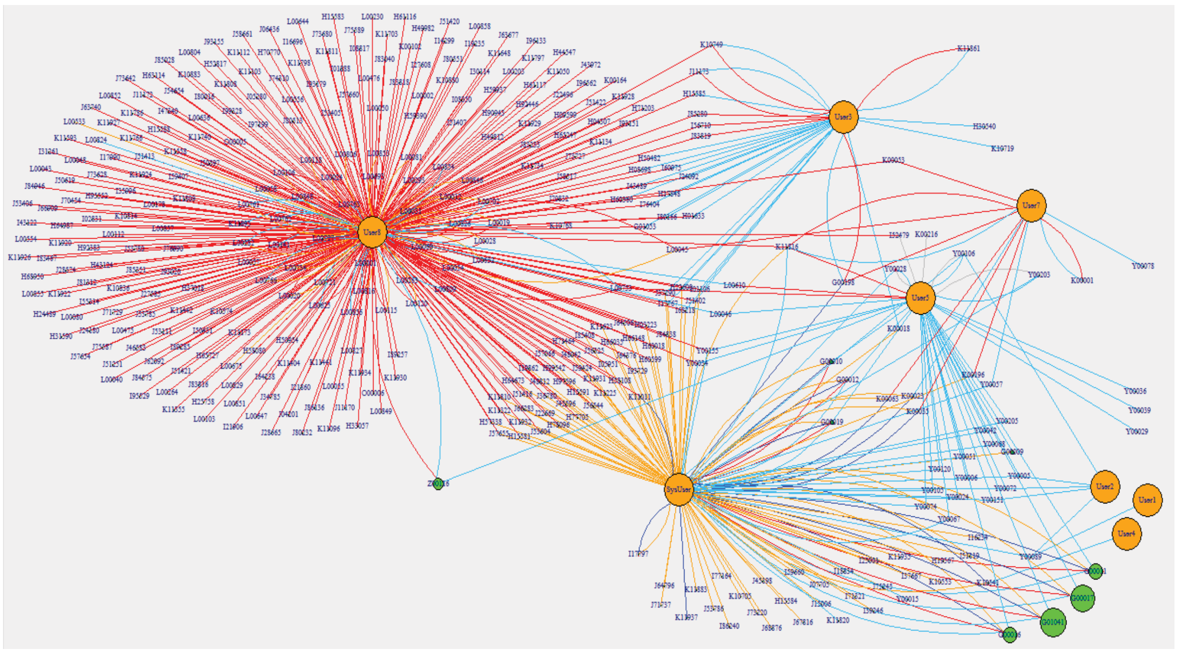


Fig. 2. Drill of the overall social network down to suppliers only related interactions

4 Conclusion and future works

To the best of our knowledge, only few research papers tackled issues in occupational fraud and even fewer integrated visual analytic concepts to their approach. Those last include unsupervised approaches like graph pattern matching techniques [Eberle et al, 2011], with strong focus on structural anomalies identification, but unfortunately forgoing real world business specificities and rules, what leads to a high rate of false positives and complex maintenance by end users. Other approaches like [Luell, 2010] or [Argyriou et al, 2013], rely on innovative but tailor made visualizations which cannot be applied to other business processes. The framework presented in this paper extends existing data mining techniques used for occupational fraud detection by offering not only visualizations to be used by auditors to uncover new fraud patterns, but also semantic reasoning capabilities for integrating those new patterns to the fraud detection engine. The targeted architecture is then scalable and extensible provided the only maintenance of specified ontologies. Our assessment of the serviceability of the sociograms on accounting journal entries delivered promising results and future directions for this research will be towards the design and the evaluation of a full prototype for supporting the framework. The generic nature of the framework presented herein and its network oriented approach also open perspectives for investigation beyond the scope of occupational fraud detection. Cyber criminality in an environment where information systems are more and more interoperable may also be investigated following a likely approach.

References

- [ACFE, 2014] ACFE 2014 Report to the nations on occupational fraud and abuse http://www.acfe.com/uploadedFiles/ACFE_Website/Content/documents/2004RttN.pdf
- [ACL, 1987] www.acl.com
- [Argyriou et al, 2013] Evmorfia N. Argyriou, Aikaterini A. Sotiraki, Antonios Symvonis. Occupational Fraud Detection Through Visualization, In Proc. of the 11th IEEE Intelligence and Security Informatics (ISI 2013), pages 4-7, 2013.
- [Chung et al., 2006] Kenneth K Chung, Liquat Hossain, Joseph Davis. Exploring sociocentric and egocentric approaches for social network analysis. KMAP 2005: Second International Conference on Knowledge Management in Asia Pacific (pp. 1-8). New Zealand: Victoria University of Wellington.
- [Cox et al., 1997] Kenneth C. Cox, Stephen G. Eick, Graham J. Wills, Ronald J. Brachman. Visual data mining: Recognizing telephone calling fraud. Data Mining and Knowledge Discovery 1997, Volume 1, Issue 2, pp 225-231.
- [Eberle et al, 2011] William Eberle - PhD, Jeffrey Graves. Insider Threat Detection Using a Graph-Based Approach, Journal of Applied Security Research, 6:32–81, 2011
- [Freeman, 2011] Linton C. Freeman. The development of social network analysis - with an emphasis on recent events. The SAGE Handbook of Social Network Analysis, SAGE Publications Ltd.
- [Csárdi et al., 2006] Gábor Csárdi, Tamás Nepusz: The igraph software package for complex network research. InterJournal Complex Systems, 1695, 2006.
- [INFLOW, 2010] http://www.orgnet.com/cases.html
- [Luell, 2010] "Employee fraud detection under real world conditions," Ph.D. dissertation, 2010. [Online]. Available: http://www.zora.uzh.ch/44863/
- [Phua et al., 2004] Clifton Phua, Vincent Lee, Kate Smith & Ross Gayler. A comprehensive Survey of Data Mining-based Fraud Detection Research. Arxiv preprint arXiv: 1009.6119.
- [PSU, 2007] https://courseware.e-education.psu.edu/courses/bootcamp/lo09/08.html
- [PWC, 2014] PriceWaterCoopers. 2014 Global economic crime survey. La fraude continue à être une vraie menace pour les entreprises.
- [Scott et al., 2011] John Scott, Peter J. Carrington. The SAGE handbook of social network analysis. SAGE Publications Ltd.
- [Wheeler et al, 2000] Richard Wheeler, Stuart Aitken. Multiple algorithms for fraud detection. Knowledge-Based Systems 13(3): 93-99.
- [Wieringa, 2009] Roel Wieringa. Design science as nested problem solving. Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology (DESRIST '09), Philadelphia, Pennsylvania, USA.
- [Yufeng et al., 2004] Yufeng Kou, Chang-Tien Lu, Sirirat Sirwongwattana, Yo-Ping Huang. Survey of fraud detection techniques. Proceedings of the 2004 IEEE. International Conference of Networking, Sensing & Control. Taipei, Taiwan, March 21-23, 2004.

Toward Better Mapping between Regulations and Operational Details of Enterprises Using Vocabularies and Semantic Similarity

Sagar Sunkle, Deepali Kholkar, and Vinay Kulkarni

Tata Research Development and Design Center
Tata Consultancy Services
54B, Industrial Estate, Hadapsar
Pune, 411013 INDIA
`{sagar.sunkle,deepali.kholkar,vinay.vkulkarni}@tcs.com`

Abstract. Industry governance, risk, and compliance (GRC) solutions stand to gain from various analyses offered by formal compliance checking approaches. Such adoption is made difficult by the fact that most formal approaches assume that a mapping between concepts of regulations and models of operational specifics exists. We propose to use Semantics of Business Vocabularies and Rules along with similarity measures to create an explicit mapping between concepts of regulations and models of operational specifics of enterprises. We believe that this proposal takes a step toward adapting and leveraging formal compliance checking approaches in industry GRC solutions.

Keywords: Regulatory Compliance, Operational Details, Business Process Models, SBVR, Semantic Similarity

1 Introduction

With non-compliance being penalized severely in most countries and across various business domains [1, 2], effective and efficient resolution of regulatory compliance is high on priority for modern enterprises. While industry governance, risk, and compliance (GRC) solutions help enterprises in managing regulatory compliance, they are mostly document-oriented and are not as rigorous as formal approaches to compliance checking. Formal compliance checking can offer several analysis benefits to industry GRC solutions such as formally finding out (non-)compliance to regulations [3–9] against document-based evidence as in industry GRC solutions, computable explanation of proofs of (non-)compliance [10, 11] against expert's judgement as in industry GRC, management of frequent changes in regulations [12, 13] against functional heat maps derived from experts' knowledge as in industry GRC, etc.

Each formal approach ideally requires to *relate* regulations to operational specifics of enterprises. A terminological mapping would essentially tell *where* in the operational activities a rule from the regulation becomes applicable. Surprisingly, formal compliance checking approaches implicitly assume such mapping to exist without describing how to arrive at it as also indicated in [14–16].

If some means were provided whereby similarity between concepts from regulations and operational specifics could be formally established, then it would be easier to *relate* concepts from regulations with operational specifics and indicate *where* a rule from regulation becomes applicable. This would also make it easier to transfer results in formal compliance checking to practical usage.

We take a step in this direction by using Semantics of Business Vocabularies and Rules (SBVR) to model vocabularies of regulations and operational specifics of enterprises. We also propose to map the concepts from structured SBVR-based vocabularies of regulations and operational specifics using semantic similarity measures.

The paper is arranged as follows. In Section 2, we review several works in formal compliance checking with regards if and how they map concepts from regulations and operational specifics of enterprise and enlist our observations. Based on our observations, we propose the use of SBVR-based vocabularies and semantic similarity measures in Section 3 to map these conceptual realms. Section 4 concludes the paper.

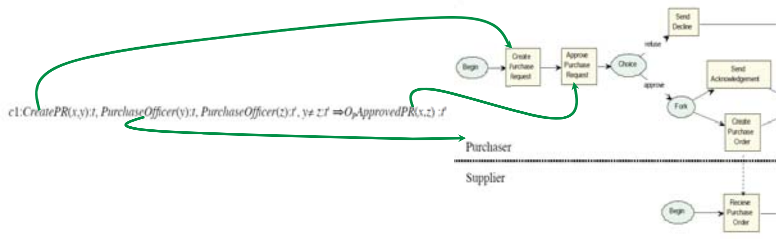
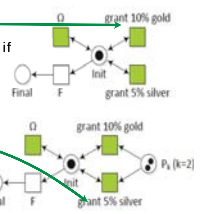
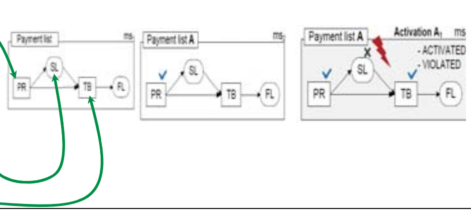
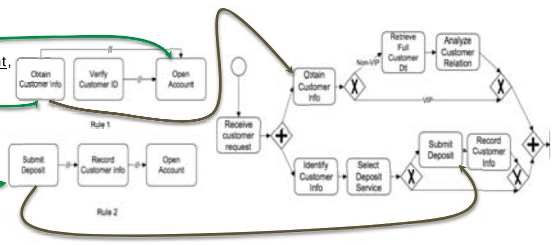
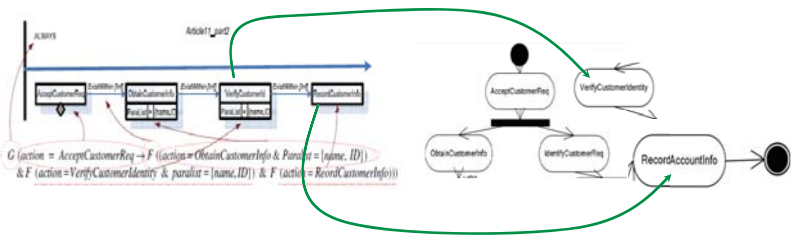
2 Related Work and Motivation

Several formal compliance checking approaches have been presented in literature. These approaches treat business process (BP) models as the de-facto representation of operational specifics of enterprise and check BP models for compliance against regulations. Our specific aim in presenting the related work is to show how these approaches *map* concepts from regulations with concepts from BP models. We consider five representative formal compliance checking approaches, namely defeasible logic-based [9, 17, 18], Petri-net based [11], compliance rule graph-based [7, 8, 19], extended BP modeling notation (BPMN) query and linear temporal logic (LTL)-based [3, 20, 21], and Business Property Specification Language (BPSL) and LTL-based approaches [22, 23]. Table 1 illustrates these approaches in two columns. First column shows how each approach maps labels/phrases from regulations to labels/phrases from approach-specific representation of BP models and second column notes formalism in that approach. In the following, we briefly elaborate the formal compliance checking approaches with regards mapping between labels/phrases row by row from Table 1.

First row from Table 1 shows defeasible logic-based approach for checking compliance of BP models against regulations [9]. Regulations are modeled in Formal Contract Language (FCL) which is a combination of efficient non-monotonic defeasible logic and deontic logic of violations. First row shows a formulation of a regulation *the creation and approval of purchase requests must be undertaken by two separate purchase officers*. Labels *CreatePR* and *ApprovedPR* from FCL expression match with *Create Purchase Request* and *Approve Purchase Request* activities from BP model respectively. Label *PurchaseOfficer* from FCL expression maps to *Purchaser* from BP model. It is evident that this mapping is presumed to exist implicitly in [9]. SBVR-based transformation of business rules to FCL expressions is suggested in [18] and semantic annotations of BP models in [17], but a structured terminological mapping of concepts is yet not explored.

Second row from Table 1 shows an approach in which an event log describing the observed operational behavior is aligned with a Petri-net pattern that formalizes a regulation. From the regulation shown in the second row of Table 1, phrases *a discount of*

Table 1: Disparity between Labels in Formal Regulations and Operational Specifics

Mapping between Regulations and Operational Details	**Compliance Rule/Operational Formalism**
 c1:CreatePR(x,y):t, PurchaseOfficer(y):t, PurchaseOfficer(z):t', y≠z:t' ⇒OApprovedPR(x,z):t'	Formal Contract Language based on Defeasible Logics; Operational Specifics as a Business Process Model
Regulation: "A discount of 10% is granted if the customer is a gold customer; 5% are granted if the customer is a silver customer." 	Rules as Petri-net Patterns, Operations from the Event Log
Regulation: "For payment runs with amount beyond euro 10,000, the payment list has to be signed before being transferred to the bank and has to be led afterwards for later audits." + Event "payment list A is transferred to the bank" 	Rules in terms of Compliance Rule Graph; Operations in terms of Events
Rule 1: Before opening an account, customer information must be obtained and verified. Rule 2: Whenever a customer requests to open a deposit account, customer information must be recorded before opening the account. 	Rules in Business Process Modeling Notation (BPMN) Query (BPMN-Q), later in Temporal Logic; operational specifics in terms of BPMN Models
	Rules in graphical Business Property Specification Language, later into Linear Temporal Logic; operational specifics in terms of BP models in the Business Process Execution Language, later into Pi calculus

10% is granted if the customer is a gold customer and *5% are granted if the customer is a silver customer* are mapped to phrases *grant 10% gold* and *grant 5% silver*. No explicit terminological mapping exists in this approach [11].

Third row from Table 1 shows an approach where events from operational event trace are checked against graph-based compliance rule language called compliance rule graph that formalizes a regulation. Phrases *payment runs*, *list has to be signed*, *transferred to the bank* from the regulation are presumed to match with similarly named events and are mapped to labels *PR*, *SL*, and *TB* respectively in the compliance rule graphs. No explicit terminological mapping has been suggested in [19] from which this example is taken or other publications from same authors [7, 8].

Fourth row from Table 1 shows an example from [3]. It uses BPMN-Q which is a visual language based on BPMN used to query BP models by matching a process graph to a query graph. Visual queries labelled Rule 1 and Rule 2 in the middle indicate BPMN-Q queries adapted to expressing the regulation on left. Interestingly, the concepts from BPMN-Q representation of the regulation match with the BP model shown by process graph on the right. This is to be expected since BPMN-Q visual queries are based on corresponding BP models. Yet, translation of regulations to BPMN-Q queries does not preserve same concepts, for instance, phrase *customer information must be obtained* is mapped to phrase *Obtain Customer Info*. Other publications by the same authors [20, 21] similarly do not express the need for explicit mapping and presume that terminological mapping from regulation statements to BPMN-Q queries exists.

Finally, fifth row from Table 1 shows an example from [22]. BP models expressed in the Business Process Execution Language are transformed into Pi calculus and then into Finite State Machines. Compliance rules captured in the graphical BPSL are translated into LTL. This way, process models can be verified against these compliance rules by means of model checking technology. The example shows that BPSL formulation of labels *RecordCustomerInfo* and *VerifyCustomerId* map to BP labels *RecordAccountInfo* and *VerifyCustomerIdentity* respectively. This approach too does not consider an explicit terminological mapping and with several transformations between specifications, lack of explicit mapping is likely to be problematic.

Table 1 essentially shows that most formal compliance checking approaches assume that labels/phrases from regulation statements map to labels/phrases used in various regulation and BP specification languages. There are approaches that recognize the need for explicit mapping between the concepts such as [16] and use word databases which consider co-occurrent words and synonyms of an activity name from the BP models. However, this approach lacks formal compliance checking as in other approaches enumerated so far. Considerable research has been done on semantic similarity of texts outside the context of regulatory compliance. An approach in [24] uses information content of texts to yield similarity judgements that correlate more closely with human assessments than other measures. Since rules and activities are short length pieces of text, it is possible to use method described in [25], which combines corpus- and knowledge-based similarity measures targeted at matching short length pieces of texts more accurately.

Industry models of operational specifics, whether they are BPMN-based BP models or enterprise data, are generally extremely large. Texts of regulations such as Sarbanes-Oxley (SOX), Foreign Account Tax Compliance Act (FATCA), BASEL-III, Dodd-

Frank and various anti money laundering regulations like Know Your Customer (KYC), etc., are similarly large with several interdependent regulations. From semantic similarity point of view, specific and short length pieces of texts need to be matched. Without such an explicit terminological mapping between these two sets of concepts, it is difficult to practically apply formal compliance checking approaches. In the next section, we sketch our proposed approach in this direction.

3 Vocabularies and Semantic Similarity

Our approach for mapping concepts from regulations and operational specifics is illustrated in Figure 1. Vocabulary$_{Reg}$ and Terminological Dictionary$_{Operations}$ indicate SBVR vocabularies of regulations and operational specifics respectively. Operational specifics may be present in any BP modeling form or as enterprise data. The concepts from individual vocabularies Vocabulary$_{Reg}$ and Terminological Dictionary$_{Operations}$ are mapped using semantic similarity measures. By expressing these concepts with a predetermined set of synonyms for each pair of concepts from both Vocabulary$_{Reg}$ and Terminological Dictionary$_{Operations}$, it is possible to express compliance checking uniformly using a given formalism. We briefly describe next how SBVR can be used to model aforementioned vocabularies.

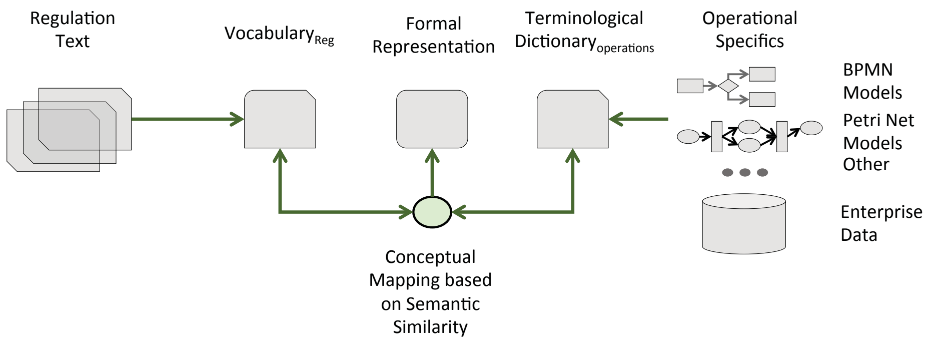


Fig. 1: Using Vocabularies and Semantic Similarity to Map Regulations and Operational Specifics

Modeling Concept Vocabularies SBVR vocabularies for regulations and operations can be defined in terms of four sections. First, vocabulary to capture the business context is created, consisting of the semantic community and sub-communities owning the regulation and to which the regulation applies. Each semantic community is unified by shared understanding of an area, i.e., body of shared meanings and a body of shared guidance containing business rules. These concepts are shown as *Business Vocabulary* in SBVR metamodel in Figure 2.

Second, the body of concepts is modeled by focusing on key terms in regulatory rules. Concepts referred in the rule are modeled as noun concepts. A general concept is defined for an entity that denotes a category. Specific details about an entity are captured as characteristics. Verb concepts capture behavior in which noun concepts play a role.

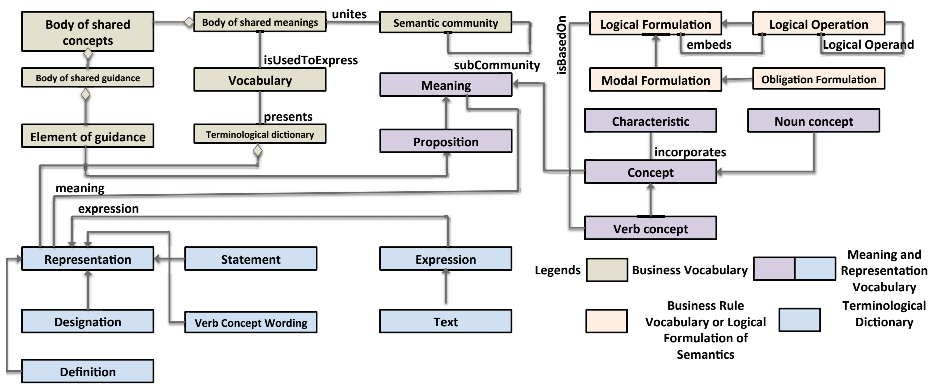


Fig. 2: SBVR Metamodel For Creating and Mapping Regulations and Operations Vocabularies

Binary verb concepts capture relations between two concepts. Characteristics are unary verb concepts. The SBVR metamodel for modeling regulation body of concepts are shown as *Meaning and Representation Vocabulary* in Figure 2.

Third, we build the body of guidance using policies laid down in the regulation. This includes logical formulation of each policy (an obligation formulation for obligatory rules) based on logical operations such as conjunctions, implications and negation. This is shown in *Business Rules Vocabulary* in Figure 2.

Fourth and lastly, we model the terminological dictionary that contains various representations used by a semantic community for its concepts and rules defined above. These consist of designations or alternate names for various concepts, definitions for concepts and natural language statements for policies stated in the regulation. We also use the terminological dictionary to capture the vocabulary used by the enterprise in its business processes. Each activity in the process becomes a verb concept wording in the terminological dictionary. SBVR concepts for modeling terminological variations are shown as *Terminological Dictionary* in Figure 2.

Mapping Concepts by Similarity Once the vocabularies of concepts from regulations and BP models are available, we can use similarity measures as in [16, 24, 25]. Note that the SBVR-based vocabularies provide a structured corpus where it is possible to encode domain knowledge including interpretations of regulations by various stakeholders [14]. The intended outcome of using vocabularies of regulations and BP models and similarity measures is that unlike approaches illustrated in Table 1, an explicit mapping between concepts of regulations and BP models will be achieved.

4 Conclusion

We illustrated several formal compliance checking approaches that assume a terminological mapping to exist between concepts of regulations and BP models when checking BP models for compliance against regulations. Structured vocabularies with semantic similarity between concepts would be needed when checking compliance of large BP models in industry against extensive regulations like BASEL-III, Dodd-Frank, FATCA, and various geography-specific KYC regulations. We plan to create an explicit mapping between regulations and BP models concept for which we briefly described how SBVR and semantic similarity measures can be used.

References

1. French Caldwell, J.A.W.: Magic quadrant for enterprise governance, risk and compliance platforms (Gartner) (2013)
2. English, S., Hammond, S.: Cost of compliance 2014 (Thomson Reuters Accelus) (2014)
3. Awad, A., Decker, G., Weske, M.: Efficient compliance checking using BPMN-Q and temporal logic. In Dumas, M., Reichert, M., Shan, M., eds.: Business Process Management, 6th International Conference, BPM 2008, Milan, Italy, September 2-4, 2008. Proceedings. Volume 5240 of Lecture Notes in Computer Science., Springer (2008) 326–341
4. Becker, J., Ahrendt, C., Coners, A., Weiß, B., Winkelmann, A.: Business rule based extension of a semantic process modeling language for managing business process compliance in the financial sector. In Fähnrich, K., Franczyk, B., eds.: Informatik 2010. Volume 175 of LNI., GI (2010) 201–206
5. El Kharbili, M., Stein, S., Markovic, I., Pulvermüller, E.: Towards a framework for semantic business process compliance management. In: The Impact of Governance, Risk, and Compliance on Information Systems (GRCIS). Volume 339 of CEUR Workshop Proceedings., Montpellier, France (June 17 2008) 1–15
6. Hashmi, M., Governatori, G., Wynn, M.T.: Normative requirements for business process compliance. In Davis, J.G., Demirkan, H., Motahari-Nezhad, H.R., eds.: Service Research and Innovation. Volume 177 of Lecture Notes in Business Information Processing., Springer (2013) 100–116
7. Knuplesch, D., Reichert, M., Ly, L.T., Kumar, A., Rinderle-Ma, S.: Visual modeling of business process compliance rules with the support of multiple perspectives. In Ng, W., Storey, V.C., Trujillo, J., eds.: Conceptual Modeling - 32th International Conference, ER 2013, Hong-Kong, China, November 11-13, 2013. Proceedings. Volume 8217 of Lecture Notes in Computer Science., Springer (2013) 106–120
8. Ly, L.T., Knuplesch, D., Rinderle-Ma, S., Göser, K., Pfeifer, H., Reichert, M., Dadam, P.: Seaflows toolset - compliance verification made easy for process-aware information systems. In Soffer, P., Proper, E., eds.: Information Systems Evolution - CAiSE Forum 2010, Hammamet, Tunisia, June 7-9, 2010, Selected Extended Papers. Volume 72 of Lecture Notes in Business Information Processing., Springer (2010) 76–91
9. Sadiq, S.W., Governatori, G., Namiri, K.: Modeling control objectives for business process compliance. In Alonso, G., Dadam, P., Rosemann, M., eds.: Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings. Volume 4714 of Lecture Notes in Computer Science., Springer (2007) 149–164
10. Antoniou, G., Bikakis, A., Dimaresis, N., Genetzakis, M., Georgalis, G., Governatori, G., Karouzaki, E., Kazepis, N., Kosmadakis, D., Kritsotakis, M., Lilis, G., Papadogiannakis, A., Pediaditis, P., Terzakis, C., Theodosaki, R., Zeginis, D.: Proof explanation for a nonmonotonic semantic web rules language. Data & Knowledge Engineering **64**(3) (2008) 662 – 687
11. Ramezani, E., Fahland, D., van der Aalst, W.M.P.: Where did I misbehave? diagnostic information in compliance checking. In Barros, A.P., Gal, A., Kindler, E., eds.: Business Process Management - 10th International Conference, BPM 2012, Tallinn, Estonia, September 3-6, 2012. Proceedings. Volume 7481 of Lecture Notes in Computer Science., Springer (2012) 262–278

12. Gómez-Sebastià, I., Álvarez-Napagao, S., Vázquez-Salceda, J., Felipe, L.O.: Towards runtime support for norm change from a monitoring perspective. In Ossowski, S., Toni, F., Vouros, G.A., eds.: Proceedings of AT 2012, Dubrovnik, Croatia, October 15-16, 2012. Volume 918 of CEUR Workshop Proceedings., CEUR-WS.org (2012) 71–85
13. Dastani, M., Meyer, J.C., Tinnemeier, N.A.M.: Programming norm change. Journal of Applied Non-Classical Logics **22**(1-2) (2012) 151–180
14. Boella, G., Janssen, M., Hulstijn, J., Humphreys, L., van der Torre, L.: Managing legal interpretation in regulatory compliance. In Francesconi, E., Verheij, B., eds.: International Conference on Artificial Intelligence and Law, ICAIL '13, Rome, Italy, June 10-14, 2013, ACM (2013) 23–32
15. Becker, J., Delfmann, P., Eggert, M., Schwittay, S.: Generalizability and applicability of modelbased business process compliance-checking approaches — a state-of-the-art analysis and research roadmap. BuR — Business Research **5**(2) (2012) 221–247 Publication status: Published.
16. Humberg, T., Wessel, C., Poggenpohl, D., Wenzel, S., Ruhroth, T., Jürjens, J.: Ontology-based analysis of compliance and regulatory requirements of business processes. In Desprez, F., Ferguson, D., Hadar, E., Leymann, F., Jarke, M., Helfert, M., eds.: CLOSER 2013 - Proceedings of the 3rd International Conference on Cloud Computing and Services Science, Aachen, Germany, 8-10 May, 2013, SciTePress (2013) 553–561
17. Governatori, G., Hoffmann, J., Sadiq, S., Weber, I.: Detecting regulatory compliance for business process models through semantic annotations. In Ardagna, D., Mecella, M., Yang, J., eds.: Business Process Management Workshops. Volume 17 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg (2009) 5–17
18. Kamada, A., Governatori, G., Sadiq, S.: Transformation of sbvr compliant business rules to executable fcl rules. In: RuleML 2010: 4th International Web Rule Symposium. Number 6403, Springer (2010) 153–161
19. Ly, L.T., Rinderle-Ma, S., Knuplesch, D., Dadam, P.: Monitoring business process compliance using compliance rule graphs. In Meersman, R., Dillon, T.S., Herrero, P., Kumar, A., Reichert, M., Qing, L., Ooi, B.C., Damiani, E., Schmidt, D.C., White, J., Hauswirth, M., Hitzler, P., Mohania, M.K., eds.: On the Move to Meaningful Internet Systems: OTM 2011. Volume 7044 of Lecture Notes in Computer Science., Springer (2011) 82–99
20. Awad, A., Polyvyanyy, A., Weske, M.: Semantic querying of business process models. In: 12th International IEEE Enterprise Distributed Object Computing Conference, ECOC 2008, 15-19 September 2008, Munich, Germany, IEEE Computer Society (2008) 85–94
21. Awad, A., Weidlich, M., Weske, M.: Visually specifying compliance rules and explaining their violations for business processes. J. Vis. Lang. Comput. **22**(1) (2011) 30–55
22. Liu, Y., Müller, S., Xu, K.: A static compliance-checking framework for business process models. IBM Syst. J. **46**(2) (April 2007) 335–361
23. Müller, S., Supatgiat, C.: A quantitative optimization model for dynamic risk-based compliance management. IBM Journal of Research and Development **51**(3/4) (2007) 295–308
24. Seco, N., Veale, T., Hayes, J.: An intrinsic information content metric for semantic similarity in wordnet. In de Mántaras, R.L., Saitta, L., eds.: Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, Valencia, Spain, August 22-27, 2004, IOS Press (2004) 1089–1090
25. Mihalcea, R., Corley, C., Strapparava, C.: Corpus-based and knowledge-based measures of text semantic similarity. In: Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA, AAAI Press (2006) 775–780

Combining Risk-Management and Computational Approaches for Trustworthiness Evaluation of Socio-Technical Systems

Nazila Gol Mohammadi[1], Torsten Bandyszak[1], Abigail Goldsteen[2], Costas Kalogiros[3], Thorsten Weyer[1], Micha Moffie[2], Bassem Nasser[4] and Mike Surridge[4]

[1]paluno - The Ruhr Institute for Software Technology, University of Duisburg-Essen, Germany
{nazila.golmohammadi, torsten.bandyszak, thorsten.weyer}@paluno.uni-due.de
[2]IBM Research Haifa, Israel
{abigailt, moffie}@il.ibm.com
[3]Athens University of Economics and Business, Athens, Greece
ckalog@aueb.gr
[4]IT-Innovation Center, School of Electronics and Computer Science, University of Southampton, Southampton, United Kingdom
{bmn, ms}@it-innovation.soton.ac.uk

Abstract. The analysis of existing software evaluation techniques reveals the need for evidence-based evaluation of systems' trustworthiness. This paper aims at evaluating trustworthiness of socio-technical systems during design-time. Our approach combines two existing evaluation techniques: a computational approach and a risk management approach. The risk-based approach identifies threats to trustworthiness on an abstract level. Computational approaches are applied to evaluate the expected end-to-end system trustworthiness in terms of different trustworthiness metrics on a concrete asset instance level. Our hybrid approach, along with a complementary tool prototype, support the assessment of risks related to trustworthiness as well as the evaluation of a system with regard to trustworthiness requirements. The result of the evaluation can be used as evidence when comparing different system configurations.

Keywords: Asset Modelling, Socio-Technical-System, Computational Evaluation, Trustworthiness Attributes, Metrics, Risk Analysis, Evaluation

1 Introduction

The technological settings in which information systems are designed, developed, and deployed have drastically changed with the advance of new technologies such as cloud computing. Socio-Technical Systems (STS) are often software-intensive information systems that interact with a variety of other software systems, as well as humans and physical entities [1, 2]. To enable designing systems with higher trustworthiness, it is crucial to analytically evaluate and estimate the trustworthiness of a system with a thorough analysis of risks and mitigation actions. This includes identifying controls to prevent threat activity at run-time. The results of this analysis should be addressed appropriately in subsequent development phases. For instance, threats identified early in the system design could yield certain trustworthiness requirements that may guide the selection or re-use of components or services. Explicit documentation of design decisions that affect the trustworthiness of a system is also essential. STS designers may need guidance when deciding whether including a certain mitigation mechanism in the system will result in an actual increase in trustworthiness, and eventually pay off. Therefore, evaluation of the End-to-End (E2E) trustworthiness of different system configurations can give some confidence in whether the detected threats are indeed prevented. Despite a large amount of literature addressing trustworthiness evaluation, the E2E evaluation of multi-faceted trustworthiness remains an open research problem. Some approaches merely focus on reliability [3] or security [4]. In contrast, our evaluation approach is based on a holistic taxonomy of software quality attributes and metrics that contribute to trustworthiness [15], including compliance, privacy, usability, complexity and many more.

We employ two complementary techniques: risk-based and computational analysis. The risk-based approach is applied at very early stage on an abstract model of the system which is independent of concrete component realizations. Complimentarily, the computational approach is performed at a later stage, on a more concrete level. It uses trustworthiness metrics, and involves calculating and aggregating them according to the complex system structures, to produce E2E metric values. Our proposed hybrid approach is a comprehensive evaluation approach that is applicable on many levels of granularity. To the best of our knowledge, no existing approaches combine these two techniques such that the consideration of threats and potential mitigations are evaluated once concrete assets are available and composed to build the system. We focus particularly on software assets that are accessible via an online marketplace with certificates holding trustworthiness metric values. Previous work aimed at establishing such a software marketplace to allow designers to select trustworthy system assets based on certificates and to compose them to create a new system [6].Our hybrid approach allows designers to evaluate a system's trustworthiness and make good design choices based on risk assessment and trustworthiness metrics. The tool support also aids designers by automatically generating software requirement documents and trustworthiness reports as evidence. The information in these documents is organized in a hierarchical way, allowing expert users to drill-down to the desired level of detail.

The remainder of the paper is structured as follows: In Section 2, we present the background and give a brief overview of existing techniques for evaluating trustworthiness. Section 3 presents our approach and evaluates it using an application example from the Ambient Assisted Living (AAL) domain. Section 4 summarizes our work.

2 Background and Related Work

This section presents the background and fundamental concepts used in our paper. We base our work on trustworthiness attributes that aim to manifest the trust concerns of end-users in an objective way. Trustworthiness attributes are quantified using trustworthiness metrics, that measure system (or individual components) behaviour, based on raw measurements, i.e., observable system properties.

Design-time models allow domain expertise to be encoded and reused in a systems' design. Such models may have different levels of abstraction. For example, an abstract system model can be used to help system designers to graphically identify and analyse the threats that can arise in a system [2], before knowing the actual deployment details. In STS, there is a particular need to explicitly specify the dependencies between assets in the system (e.g., host of application). An Asset is anything of value in an STS [7, 8], including software, hardware and humans. A Workflow model specifies a set of concrete asset instances, as well as their interrelations. A system can be described using several such workflows, each representing a certain process (or use case) performed by the system.

There are many standards and methods [9, 10, 11, 12] that describe the risk management methodology and provide support in the process of identifying the system assets and relevant threats. The CORAS project [13] aimed at simplifying the task using a graphical approach to identify, explain and document security threats and risk scenarios. However, these approaches depend on humans. Microsoft's SDL threat modelling tool provides a graphical user interface for developers to generate threat models based on software architecture diagrams. However, Microsoft's SDL threat modelling tools may be more likely to overlook threats beyond the scope of STRIDE [14] (e.g., human-centred attacks) unless they also involve security professionals.

The need to evaluate the overall trustworthiness of a system has been recognised by several researchers. Elshaafi et al. [15] present an approach towards measuring the trustworthiness of a service composition focusing on run-time monitoring and targeting reputation, reliability, and security considering several service compositions. Similarly, Zhao et al., [16] propose a framework for trustworthy web service management, which aggregates the availability, reliability, and response time of services composed in sequence, parallel, conditional, and loop structures. Other approaches, such as [17], focus on reputation only by aggregating service ratings in order to determine the provider's trustworthiness. Cardoso et al., [18] utilize graph reduction mechanisms and respective formulas for aggregating time, cost, and reliability of service workflows. Hwang et al. [19] propose a probabilistic approach for estimating certain quality of service of respective compositions. While the above approaches support a large number of composition patterns, they focus on a limited set of trustworthiness metrics. Closer to our approach is the work of Jaeger et al. [20] where they support a wider set of QoS metrics.

3 Trustworthiness Evaluation of Socio-Technical Systems

This section describes our trustworthiness evaluation approach. First, a conceptual model of our approach is presented. Second, we describe how we combine two existing techniques at two different abstraction levels.

3.1 Overview of Our Approach

Meta-model for Design-Time Trustworthiness Evaluation. The fundamental concepts and their relations are depicted in Fig. 1.

We distinguish between *Asset Categories,* generic types of system building blocks on an abstract level, and *Asset Instances,* concrete instances pertaining to a certain category, e.g., a concrete software application or implementation. A *Threat* is a situation or event that, if active, could undermine the value of an asset by altering its behaviour. *Controls* are trustworthiness requirements that aim at mitigating threats.

For computational evaluation of trustworthiness in our approach, *Metrics* are used as functions to quantify system trustworthiness. A Metric is a standard way for measuring and quantifying certain trustworthiness attributes and more concrete quality properties of a system [5]. Metric values of a specific Asset instance are provided within a trustworthiness certificate that is often provided together with the software itself on a marketplace.

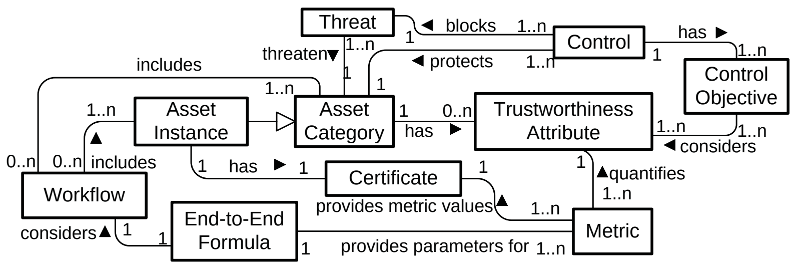


Fig. 1. Conceptual Model for Trustworthiness Evaluation

A *Certificate* is created by a certification authority that evaluates a software system or asset in order to confirm that it meets some trustworthiness goals. It describes all observed trustworthiness properties of the software, as well as related evidence in terms of certified metric values. In order to enable trustworthiness computation for a whole *E2E* system configuration, and thereby aggregate the certified metric values for each of its asset instances, E2E formulas are required. Since *Workflows* describe the concrete instance relations, each E2E formula is particular to a certain Workflow.

Combining Risk Assessment with Computational Approaches toward Trustworthiness Evaluation. In the proposed E2E trustworthiness evaluation, the risk-based approach is performed on the level of asset categories while the trustworthiness metric computation is based on metric values of asset instances (illustrated in Fig. 2). Trustworthiness evaluation starts with a design-time system model which describes the general building blocks of an envisioned STS in an abstract level. It includes both physical, and logical assets (e.g., software), as well as humans that interact with the system. This model is independent of concrete realizations, i.e., without considering which asset instances that shall be deployed as implementations of software assets.
At this early stage our approach already allows us to identify threats to correct system behaviour at run-time. To this end, a knowledge base of threats is used to identify relevant threats to the asset based on its type (e.g., logical asset, physical asset, etc.) and its relations patterns (e.g., client-server relation). Given the threats and potential

controls the designer is provided with a statement on the risks and corresponding actions that can be taken or at least planned for at design-time. Based on this information, asset structures may be revised, or informed decisions on the concrete asset instances can be made.

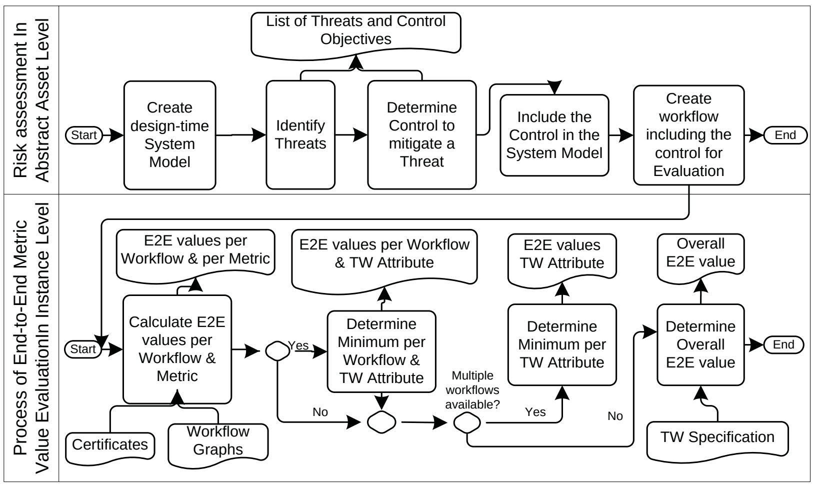

Risk assessment In Abstract Asset Level
List of Threats and Control Objectives
Start
Create design-time System Model
Identify Threats
Determine Control to mitigate a Threat
Include the Control in the System Model
Create workflow including the control for Evaluation
End
Process of End-to-End Metric Value EvaluationIn Instance Level
E2E values per Workflow & per Metric
E2E values per Workflow & TW Attribute
E2E values TW Attribute
Overall E2E value
Start
Calculate E2E values per Workflow & Metric
Yes
Determine Minimum per Workflow & TW Attribute
Multiple workflows available?
Determine Minimum per TW Attribute
Determine Overall E2E value
End
No
Yes
No
Certificates
Workflow Graphs
TW Specification

Fig. 2. Trustworthiness evaluation in two different abstraction layers

Once concrete instances (e.g., available on a marketplace) of the asset categories are selected and modelled in terms of one or multiple workflows, the designer can then use the computational trustworthiness evaluation approach to calculate E2E trustworthiness based on the certified metric values of each of the asset instances, and the E2E formulas for each relevant workflow.

The system structure needs to be considered and reflected in the E2E value. We considered different component structures for determining an "E2E" trustworthiness value based on metrics. Redundancy structures, which are defined as a means to assure correct system performance and thereby increase trustworthiness levels, were especially considered in the E2E trustworthiness calculation. The explicit description of respective metrics is a precondition for the calculation of E2E trustworthiness value, which requires certified metric values of each involved asset as parameters.

The resulting E2E trustworthiness values provide detailed information about the aggregated trustworthiness of the asset instances that are involved in a certain workflow. This allows the designer to relate the threats to the affecting trustworthiness values, and also evaluate and substantiate the effectiveness of applied design-time controls. Assuming that the metric values reflect the existence of controls (e.g., confidentiality, authentication), then the quantification enhances the evaluation. In order to facilitate the interpretation of the calculated E2E trustworthiness, the initial values that are particular to a certain workflow can again be aggregated so that finally one value for the whole system trustworthiness can be obtained. To this end, for instance, a pessimistic approach can be followed, and the minimum of two or more metrics per attribute, or workflows can be calculated. A required precondition is that the value ranges of different metrics are comparable.

3.2 Application Example

This section describes an application example for demonstrating our two-fold approach on different levels of abstraction (illustrated in Fig. 5).
The example scenario focuses on a Fall Management System (FMS) from the AAL domain. FMS allows elderly people in their homes to call for help in case of emergency situations. These emergency incidents are reported to an Alarm Call centre that, in turn, reacts by e.g., dispatching ambulances or other medical caregivers, e.g. the relatives. The starting point for evaluating the trustworthiness of such an STS is a design-time system model (depicted in Fig. 3). An elderly uses a *Personal Emergency Response System* (PERS) device to call for help, which is then reported to the Alarm Call Center that uses an *Emergency Monitoring and Handling Tool* (EMHT) to visualize, organize, and manage incidents. Hence, the EMHT is a software service hosted by the Alarm Call Center operated by a Healthcare authority. Emergency notification and Ambulances Services, which are run on mobile phones of relatives, or by Ambulance Stations respectively, are called in order to require caregivers to provide help.

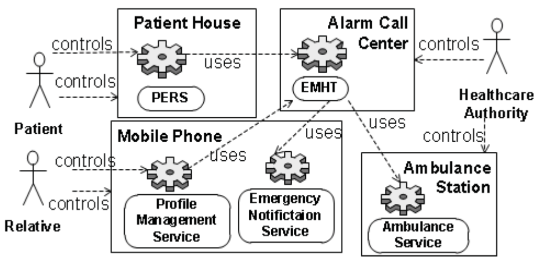


Fig. 3. Design time System Model of the Fall Management System Specifying Asset Categories

Identification of Threats and Controls. Based on the design-time system model, which specifies the relevant asset types of the system, and their relations, the trustworthiness evaluation is first performed on this abstraction layer of abstract assets. The "Evaluate System E2E Trustworthiness" use case of our prototype tool supports the designer in this task. The model file is passed to the System Analyser for analysing the threats that may affect each system asset. The System Analyser reports for each asset type the related threats as well as the potential controls that can be applied to prevent or mitigate the threats. For instance, a threat that may arise at run-time is the unavailability of the EMHT asset. This will probably lead to a failure of the whole STS, since the EMHT is a central service that enables and facilitates handling incoming calls for help. Hence, a possible control to react to this threat at run-time may be service substitution, i.e., to switch to another backup service that may also be a different implementation of the asset category. In order to address and implement this control at design-time, two or more concrete EMHT realizations have to be considered as redundant instances of the EMHT asset. The identified control will be considered (including redundant asset instances) and modelled as a workflow.

Trustworthiness Calculation for Asset Instance Configurations. As a required precondition for E2E trustworthiness calculation, the designer has to select the evaluation criteria to be used, i.e., the weights of relevant trustworthiness attributes to the overall E2E TW. The weights represent the designer's preferences regarding the relevance of each trustworthiness attribute. The designer continues by uploading one or more workflows. Fig. 4 shows an exemplary workflow for our FMS example. The designer specifies the redundant asset instances EMHT_1 and EMHT_2 and Ambulance_Service_1 to 3 for the asset types "EMHT" and "Ambulance_Service" respectively. Based on the Workflow graphs, the Formula Builder of the E2E TWE tool will create formula skeletons for any kind of metrics.

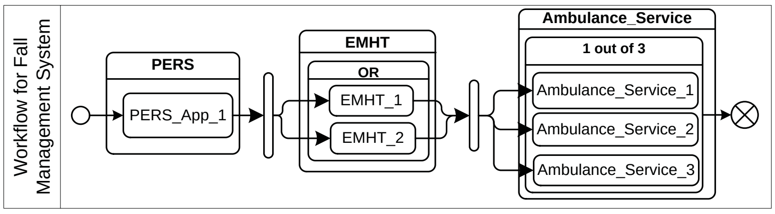


Fig. 4. Example Workflow of the Fall Management System

These skeletons will be combined according to the sequence structures of the asset categories modelled in the workflows. For all of the existing asset instances, certificates containing Metric values stored as evidences must be provided. The E2E Trustworthiness Calculator will extract the metric values from the certificates and use them in the E2E computation. Here, the attribute weights that have been specified by the designer in the first place will be used in order to obtain a single trustworthiness values for the whole system composition.

These relations between the two complementary approaches are illustrated in Fig. 5, which also shows how the generated output supports the evaluation.

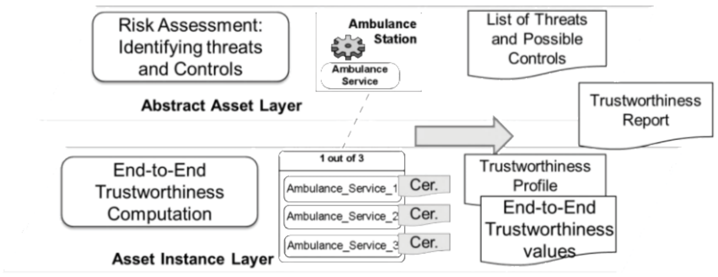


Fig. 5. Example relation between the complementary approaches for trustworthiness evaluation

4 Summary

This paper addresses the problems of the commonly used techniques for evaluating overall trustworthiness in STS. We suggest a combination of two complementary techniques: computational approach and risk management approach. Threats and controls proposed by the risk based approach can be evaluated against actual trustworthiness values, thus substantiating the effectiveness of applied design-time controls. We also presented a tool prototype that supports the assessment of risks related to trustworthiness as well as the evaluation of a system with regard to trustworthiness requirements, aiding the designer in making trustworthiness related decisions and providing evidence and documentation for those decisions.

References

1. Sommerville, I.: Software Engineering, 9th edition, Addison-Wesley, 2011.
2. Lock, R., Sommerville, I.: Modelling and Analysis of Socio-Technical System of Systems, In: 15th IEEE Int'l. Conference on Engineering of Complex Computer Systems, 2010.
3. Dev G. Raheja, Louis J. Gullo: Design for Reliability. Wiley, 2012.
4. Avizienis, A., Laprie, J.-C., Randell, B. and Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing, IEEE Trans. on Dependable and Secure Computing 1 (1), 11-33, 2004.
5. Gol Mohammadi, N., Paulus, S., Bishr, M., Metzger, A., Koennecke, H., Hartenstein, S., Weyer, T. and Pohl K.: Trustworthiness Attributes and Metrics for Engineering Trusted Internet-based Software Systems. In: Cloud Computing and Services Science, (CLOSER Selected Paper), Communications in Computer and Information Science 453. 19-35, 2014.
6. Ali, M., Sabetta, A., Bezzi, M.: A Marketplace for Business Software with Certified Security Properties, In: Cyber Security and Privacy Communications. Computer and Information Science 182. 105-114, 2013.
7. Gol Mohammadi, N., Bandyszak, T., Moffie, M., Chen, X., Weyer, T., Kalogiros, C., Nasser, B., and Mike Surridge: Maintaining Trustworthiness of Socio-Technical Systems at Run-Time, In: Proceedings 11th Int'l. Conference TrustBus, 2014
8. Surridge, M., Nasser, B., Chen, B., Chakravarthy, A., and Melas, P.: Run-Time Risk Management in Adaptive ICT Systems, In: 8th Int'l. Conference on Availability, Reliability and Security (ARES), 102,110, 2013.
9. Christopher, J. A. and Dorofee, A., Managing Information Security Risks: The Octave Approach. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
10. Fray, I.L.: A comparative study of risk assessment methods, MEHARI & CRAMM with a new formal model of risk assessment (FoMRA) in information systems. In Proc. of the 11th Int'l. Conference on Computer Information Systems and Industrial Management, 2012.
11. ISO/IEC 27005, Information technology — Security techniques — Information security risk management, https://www.iso.org/obp/ui/#iso:std:iso-iec:27005:ed-2:v1:en, 2011
12. OWASP.org (2013), https://www.owasp.org/index.php/Top_10_2013-Top_10 , 2013
13. Hogganvik, I. and Stølen, K.: A graphical approach to risk identification, motivated by empirical investigations, In: Proceedings of the 9th Int'l. Conference on Model Driven Engineering Languages and Systems (MoDELS'06), 2006.
14. Swiderski, F. and Snyder, W.: Threat Modelling. Microsoft Press. 2004.
15. Elshaafi, H., McGibney, J. & Botvich, D. Trustworthiness monitoring and prediction of composite services. Computers and Communications (ISCC), 2012 IEEE Symposium on (pp. 000580–000587), 2012.
16. Zhao, S., Wu, G., Li, Y., and Yu, K.: A Framework for Trustworthy Web Service Management. In: 2nd Int'l. Symposium Electronic Commerce and Security, 479-482, 2009.
17. Malik, Z., and Bouguettaya, A.: RATEWeb: Reputation Assessment for Trust Establishment among Web services. In: VLDB Journal, Vol. 18, Issue 4, pp. 885-911, 2009
18. Cardoso, J., Sheth, A., Miller, J., Arnold, J., and Kochut, K.: Quality of Service for Workflows and Web Service Processes. In: Web Semantics: Science, Services and Agents on the World Wide Web 1 (3), 281-308, 2004.
19. Jaeger, M. C., Rojec-Goldmann, G. and Mühl, G.: QoS Aggregation for Web Service Composition using Workflow Patterns. In: Proceedings of the 8th IEEE Int'l. Enterprise Distributed Object Computing Conf (EDOC 2004), pp. 149 – 159, 2004
20. Hwang, S. Y., Wang, H., Tang, J., Srivastava, J.: A Probabilistic Approach to Modeling and estimating the QoS of web-services-based workflows. In: Journal of Information Sciences 177, 5484–5503, 2007