Martin Glinz
Patrick Heymans (Eds.)

LNCS 5512

Requirements Engineering: Foundation for Software Quality

15th International Working Conference, REFSQ 2009
Amsterdam, The Netherlands, June 2009
Proceedings

Lecture Notes in Computer Science 5512

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Martin Glinz Patrick Heymans (Eds.)

Requirements Engineering: Foundation for Software Quality

15th International Working Conference, REFSQ 2009
Amsterdam, The Netherlands, June 8-9, 2009
Proceedings

Springer

Volume Editors

Martin Glinz
University of Zurich
Department of Informatics
Binzmühlestrasse 14, 8050 Zurich, Switzerland
E-mail: glinz@ifi.uzh.ch
www.ifi.uzh.ch/~glinz

Patrick Heymans
FUNDP–University of Namur
Computer Science Faculty
PReCISE Research Centre
Rue Grandgagnage 21, 5000 Namur, Belgium
E-mail: phe@info.fundp.ac.be
www.info.fundp.ac.be/~phe

Library of Congress Control Number: Applied for

CR Subject Classification (1998): D.2, D.2.1, D.2.3, D.1, I.2.2, D.1.2

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743
ISBN-10 3-642-02049-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-02049-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12688807 06/3180 5 4 3 2 1 0

Preface

This volume contains the papers accepted for presentation at the 15th Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2009), held in Amsterdam during June 8–9, 2009.

Since 1994, when the first REFSQ took place, requirements engineering (RE) has never ceased to be a dominant factor influencing the quality of software, systems and services. Initially started as a workshop, the REFSQ working conference series has now established itself as one of the leading international forums for discussing RE in its many relations to quality. It seeks reports on novel ideas and techniques that enhance the quality of RE products and processes, as well as reflections on current research and industrial RE practices.

One of the most appreciated characteristics of REFSQ is that of being a highly interactive and structured event. Each session is organized in order to provoke discussion among the presenters of papers, discussants and all the other participants. Typically, after a paper is presented, it is immediately discussed by one or two pre-assigned discussants, then subject to a free discussion involving all participants. At the end of each session, an open discussion of all the papers presented in the session takes place. REFSQ 2009 maintained this tradition.

The special theme of REFSQ 2009 was *value* and *risk* in relation to RE and quality. Ensuring that requirements, and eventually running systems, meet the values of the individuals and organizations that they are meant to serve has always been at the core of RE. Nowadays, continuously changing technology, ubiquitous software, ever-growing system complexity, and unheard of market pressure, simultaneously with new business models based, for example, on crowdsourcing, make the concern for value all the more present and challenging. The notion of value is inseparably connected to the notion of risk. We are challenged both by product risks, i.e., risks that threaten the value we want to achieve with the systems we build, and project risk, i.e., the risk of not achieving the intended value when building a system. Identifying and mitigating risks is a core task of RE.

While REFSQ 2009 invited general submissions on RE, papers dealing with value and risk were especially welcome. In all, we received a healthy 60 submissions, consisting of 49 full papers and 11 short papers. After all the submissions were carefully assessed by three independent reviewers and went through electronic discussions, the Program Committee met and finally selected 14 top-quality full papers (11 research papers and 3 experience reports), resulting in an acceptance rate of 29% (14/49) for full papers. In addition to those 14 papers, 7 high-quality short papers were selected: 4 were shortened versions of very promising but not fully mature long papers, while the remaining 3 were selected from the 11 submitted short papers. The overall acceptance rate of the conference was thus of 35% (21/60).

The table of contents shows a topical grouping of papers highlighting the success of this year's special theme. It also indicates consistent interest of the community in already popular topics such as change and evolution, inconsistency, interaction, structuring, elicitation, creativity, documentation, modelling and research methods. The work presented at REFSQ 2009 continues to have a strong anchoring in practice with empirical investigations spanning over a wide range of application domains, including embedded systems (automotive, mobile communication, navigation), off-the-shelf and open-source sofware, business and Web-based information systems, smart homes, and product lifecycle management. The international character of REFSQ is underlined by the 12 countries represented[1] this year with a notable 33% (7/21) of papers coming from North America. Table 1 provides more details[2].

Table 1. Authors and papers per country

Country	Number of papers	Number of authors
Australia	1	1
Austria	1	2
Belgium	1	4
Canada	1	4
France	2	5
Germany	4	10
Greece	1	1
Spain	1	2
Sweden	5	12
Switzerland	1	3
UK	4	10
US	6	10

As in previous years, these proceedings serve as a record of REFSQ 2009, but also present an excellent snapshot of the state of the art of research and practice in RE. As such, we believe that they are of interest to the whole RE community, from students embarking on their PhD to experienced practitioners interested in emerging knowledge, techniques and methods.

All readers who are interested in an account of the discussions that took place during the conference should consult the post-conference summary that we intend to publish as usual in the *ACM SIGSOFT Software Engineering Notes*.

REFSQ is essentially a collaborative effort. First of all, we thank Anne Persson and Guttorm Sindre, who served REFSQ 2009 very well as Organization Chairs. We are also indebted to the CAiSE 2009 organization team for their efficient support in co-locating both events. REFSQ would not be in its 15^{th}

[1] Computed from the affiliations of the authors.

[2] The sum of the second column is higher than the total number of accepted papers because some papers have authors from multiple countries.

edition this year without the wise guidance of our Advisory Board – Eric Dubois, Andreas L. Opdahl and Klaus Pohl – as well as all the former conference chairs.

As the Program Chairs of REFSQ 2009, we deeply thank the members of the REFSQ 2009 Program Committee and the additional referees for their careful and timely reviews. We especially thank those who actively participated in the Program Committee meeting and those who volunteered to act as shephards to help finalize promising papers. Finally, we would like to thank our collaborators at the University of Zurich and the University of Namur: Eya Ben Charrada, Cédric Jeanneret, Reinhard Stoiber and Tobias Reinhard for their careful format checking of the camera-ready submissions, Germain Saval and Andreas Classen for their support in setting up the website and for the layout of the call for papers. Last, but not least, we thank Evelyne Berger and Laura Oger who provided all sorts of administrative support.

April 2009

Martin Glinz
Patrick Heymans

Organization

Erik Kamsties	University of Applied Sciences, Lübeck, Germany
Søren Lauesen	IT University of Copenhagen, Denmark
Seok-Won Lee	University of North Carolina at Charlotte, USA
Michel Lemoine	ONERA, France
Nazim H. Madhavji	University of Western Ontario, Canada
Neil Maiden	City University, London, UK
Raimundas Matulevičius	University of Namur, Belgium
Haris Mouratidis	University of East London, UK
John Mylopoulos	University of Toronto, Canada
Cornelius Ncube	Bournemouth University, UK
Bashar Nuseibeh	Open University, UK
Barbara Paech	University of Heidelberg, Germany
Oscar Pastor	Valencia University of Technology, Spain
Colette Rolland	Université Paris I – Panthéon Sorbonne, France
Gil Regev	EPFL, Switzerland
Björn Regnell	Lund University, Sweden
Camille Salinesi	Université Paris 1 - Panthéon Sorbonne, France
Kristian Sandahl	Linköping University, Sweden
Peter Sawyer	Lancaster University, UK
Kurt Schneider	University of Hannover, Germany
Janis Stirna	Jönköping University, Sweden
Axel van Lamsweerde	Université Catholique de Louvain, Belgium
Roel Wieringa	University of Twente, The Netherlands
Eric Yu	University of Toronto, Canada
Didar Zowghi	University of Technology Sydney, Australia

Additional Referees

Lars Borner	Mahvish Khurum
Jordi Cabot	Kim Lauenroth
Oscar Dieste	Armstrong Nhlabatsi
Golnaz Elahi	Zornitza Racheva
Anna Grimán	Jürgen Rückert
Jennifer Horkoff	Henk van der Schuur
Siv-Hilde Houmb	Christer Thörn
Slinger Jansen	Inge van de Weerd
Marijke Janssen	Krzysztof Wnuk
Sybren de Kinderen	

Sponsoring Institutions

- University of Namur, Belgium
- University of Zurich, Switzerland
- CAiSE 2009 conference organization
- Interuniversity Attraction Poles Programme, Belgian State, Belgian Science Policy (MoVES project)

Table of Contents

5. Experience

6. Elicitation

7. Research Methods

8. Behavior Modeling

9. Empirical Studies

10. Open-Source RE

When Product Managers Gamble with Requirements: Attitudes to Value and Risk

Nina D. Fogelström[1], Sebastian Barney[1], Aybüke Aurum[2], and Anders Hederstierna[3]

[1] School of Engeering, Blekinge Institute of Technology
nino.dzamashvili.fogelstrom@bth.se, sebastian.barney@bth.se
[2] School of Information Systems, Technology and Management, University of New South Wales
aybuke@unsw.edu.au
[3] School of Management, Blekinge Institute of Technology
anders.hederstierna@bth.se

Abstract. [**Context and motivation**] Finding a balance between commercial (customer specific, market pull and external quality requirements) and internal quality requirements is a recognized challenge in market driven software product development (MDSPD). In order to address this challenge it is important to understand the preferences and biases influencing decision makers selecting requirements for software releases. [**Question/problem**] Prospect theory has been successfully applied to many disciplines. Applying it to MDSPD suggests decision makers will avoid risk when selecting between commercial requirements, take risk with internal quality requirements, and prefer commercial requirements over internal quality requirements in order to maximize their perceived value. This paper seeks to investigate this claim. [**Principal ideas/results**] This paper presents an experiment investigating whether the biases proposed by prospect theory can be seen operating in MDSPD requirements engineering (RE). The results indicate risk avoidance when dealing commercial requirements, while greater risk is taken when dealing with internal quality requirements. [**Contribution**] As this is the first paper to use prospect theory to explain requirements selection decisions, it presents opportunity to educate people in the biases they bring to the RE process, and facilitate the creation of strategies for balancing the different requirements types.

Keywords: requirements selection, prospect theory, value, risk.

1 Introduction

Requirements engineering (RE) is a decision rich activity, and RE decision support as a field has received increasing attention as market-driven software product development (MDSPD) has taken off [1,2,3]. Due to the growing popularity and complexity of MDSPD, it is important to understand how inherit biases influences decisions made in this area. This paper explores one key bias presented through prospect theory, and how it impacts value and risk perception.

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 1–15, 2009.

MDSPD is focused on the task of selecting a set of requirements that should be included in the coming releases of a product. The success of the product is measured in sales and is related to how much the product features are valued by the customers.

The requirements selection process in MDSPD is often perceived as a very complex activity. This complexity is explained by:

- A large number of requirements that need to be considered;
- A variety of different technical aspects that need to be taken into account before a selection of the requirements can be made; and
- The challenge associated with taking decisions based on the decision material of variable quality (such as uncertain cost and value estimations).

Thus, the authors of this paper classify requirements selection as occurring in a high-risk environment.

Requirements selection in MDSPD often involves situations where a decision maker must choose between requirements of different types. Key requirement types include *commercial requirements* – originating from the market and key customers; and *system requirements* (internal quality requirements) – often connected with system maintenance, quality and architecture. The challenge here is to find a balance between commercial and system requirements, such as to allow satisfying the immediate market and customer needs, as well as assuring the healthy product architecture evolution.

Recent studies on MDSPD show that commercial requirements are perceived more important than system requirements [4,5,6,7]. However, these studies also show that system requirements are generally undervalued. Considering the impact of requirements selection decisions on the business of a development company, it would be useful to have an understanding of the heuristics and cognitive biases behind these decisions, which according to the classical decision theory govern the decision outcome [8].

In this paper prospect theory [9,10] is used to explore biases behind requirement selection decisions in a MDSPD context. While this breaks normative assumptions about decision making in risky situations referenced in most engineering literature [11], decisions made by software project managers have found to be better modelled by prospect theory than the more common utility theory.

Prospect theory models decision makers' attitudes towards risk in terms of gains and losses [8]. It predicts that people will be risk adverse when selecting between options described in terms of gains, and risk taking when deciding between options described in terms of losses. Given that software requirements in MDSPD RE can be described in terms of revenues (gains) and/or costs (losses) it seems interesting to apply prospect theory in order to explain the choices of decision makers.

Assuming that in MDSPD commercial requirements are normally associated with revenue, while system requirements are associated with costs, applying prospect theory would suggest decision makers will take risks with system requirements. As these requirements are associated with cost, the preference is to

delay spending in the short-term, despite the potential for increased costs later and thus taking a risk.

Conversely, since commercial requirements are associated with gains, decision makers will be more risk adverse when selecting and prioritizing these requirements. This means preference will be given to requirements with a more reliable revenue stream, even if the alternatives could yield greater ROI.

In the experiment presented in this paper the authors investigate how well prospect theory applies to MDSPD RE. To the best of the author's knowledge this is the first time prospect theory has been applied to MDSPD. The primary objective is to investigate the decision maker's attitude towards risk, as well as perception of requirements value when it comes to choosing between alternatives of both cost and benefit.

The structure of the paper is as follows; Section 2 presents the special characteristics of MDSPD and provides an overview of how prospect theory can be used to explain and model requirements selection decisions in an MDSPD environment; Section 3 details the research question and the experiment design; the experiment results and analysis are found in Section 4 and finally Section 5 presents conclusions and future work.

2 Background

The following section provides a brief summary of the characteristics of requirements selection process in MDSPD. This is followed by a presentation of prospect theory and an experiment that explore attitudes towards losses and gains. The section concludes with a discussion on how prospect theory can be used to explain requirements selection decisions in MDSPD.

2.1 Market Driven Software Product Development

In typical MDSPD the development organization is the main stakeholder and owner of a developed product. The product evolves over time with new functionality added and offered to the general market through product releases. In MDSPD the development organization takes most risks and is responsible for development costs of the product. Delivering the right set of functionality when the market is ready for it is critical for the success of the software product.

Market-driven development is largely product focused and many important activities are performed prior to the initiation of the development projects (i.e. pre-project). The goal of pre-project activities is to catch, analyse, select and plan requirements for future releases of the product. Pre-project activities should result in a prioritized list of requirements that are assigned to a specific release. These requirements are then realized through one or more development projects.

The importance of correct decision making pre-project is acknowledged and highlighted by both industry and academia. However, finding a perfect set of requirements is considered impossible due to a variety of different criteria that need to be taken into account and conflicting interests of stakeholders. Each stakeholder group is interested to see their requirement in the next release [12,13,14].

Table 1. Requirement Types in MDSPD

Requirement Type	Explanation
Customer specific features	Requirement originating from one of the key customers
Market pull requirements	Requirements originating from more than one customers that represent a certain market
Innovation	Requirement usually originating from the company's R&D. Examples of this may be innovative technology, patents, innovative features, etc.
External quality aspects	Requirements concerning usability, security, reliability and other quality issues
System requirements (Internal quality aspects)	These are system architecture specific requirements, connected to system architecture health, flexibility, and evolution.

Different requirement types that are typically involved in requirements selection process are shown in Table 1 [2,15,16]. *Customer specific features* and *Market pull requirements*, which are often referred to as *commercial requirements* originate from the customers. In the case of *Customer specific features* the main stakeholder is one key customer, whereas for *Market pull requirements* the main stakeholder is often a marketing or sales unit representing a number of customers.

Innovation requirements can originate from *Key customers*, however mostly these requirements are defined by the R&D unit of a company.

The stakeholders of *External quality aspects* can be represented by both external parties, such as customers and internal parties such as R&D. Finally, *System requirements* mainly focus on system architecture and evolution aspects and almost always originate from organisations' R&D units.

Recent studies on MDSPD show that commercial requirements are perceived more important than system requirements [4,5,6,7]. However, these studies also show that system requirements are generally undervalued. The situation where system requirements are usually given a lower priority is troubling system engineers as it will lead to system architecture deterioration and collapse in the longer term, which can be connected to serious losses for the company.

Finding a trade-off and profitable balance between commercial and system requirements is very important for the success of a product. But before this trade-off can be found, the authors believe an understanding should be reached of the reasoning and decision frames that govern decision making process in MDSPD RE. This is where the authors believe prospect theory may be useful.

2.2 Prospect Theory and Related Works

Prospect theory was developed by Daniel Kahneman and Amos Tversky [9], for which Kahneman won a Nobel Prize in economics in 2002. It examines possible outcomes in terms of gains and losses to determine the value of these outcomes [8]. When asked about the preferred choice between the "sure thing"

Table 2. Risk Preferences

Scenario	Risk Attitude	Risk Preference
$V(X) = V(0.5 * 2X)$	Risk neutral	Indifferent between the options
$V(X) > V(0.5 * 2X)$	Risk adverse	Prefers the "sure thing"
$V(X) < V(0.5 * 2X)$	Risk taker	Prefers the risky option

and a risky option, it is assumed that the decision maker's attitude to risk is revealed. If X is the monetary outcome, 0.5 is the probability of "winning" X and V is the value of a choice option, the preferences in Table 2 reveal the risk attitude of an individual.

While prospect theory is built on utility theory, prospect theory differs in that it considers issues from within a reference frame. Utility theory looks at net wealth, where prospect theory considers gains and losses. Thus prospect theory can explain why a loss of $500 is felt more strongly than a gain of $500.

Prospect theory has many practical applications, even within software engineering. It has been used to understand the reasons behind software project escalation [17], explain why early involvement in software project bidding leads to higher quotes [18], and support with the pricing of software bundling [19]. What is interesting about prospect theory is the relationship between the outcome and value is not linear, as seen in Figure 1. The function is steepest around zero and flattens out as the losses and gains become greater. It is also steeper for losses than gains.

The properties of the function mean that $500 loss is felt more strongly than a $500 gain as the relative value for losses is greater than the value for a similar sized gain. This can be seen in Figure 1 by using the guide markings.

Given the choice between (a) a 50% chance of gaining $1000 or (b) a sure gain of $500, most people select the safer option, B [9]. Figure 1 shows that a

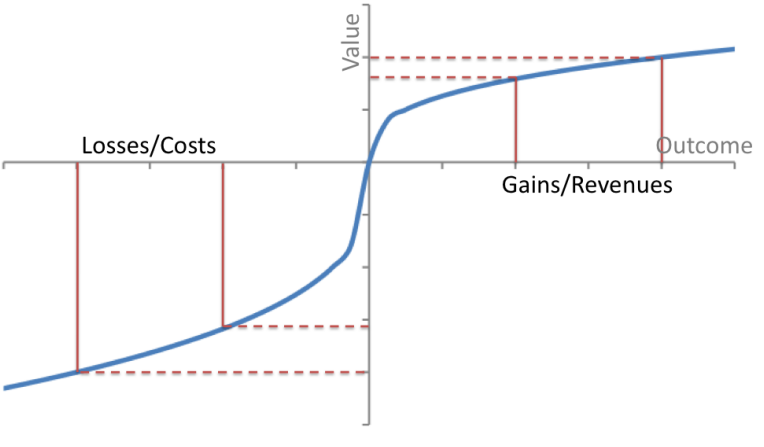


Fig. 1. Value function from prospect theory

doubling in gains is not equivalent to a doubling in value, so the value delivered by option B on average is actually less than option A (as the value derived from the average of no gain and a $1000 gain is less than the value of a $500 gain).

Conversely when presented with options to (a) lose $1000 with 50% probability or (b) a sure loss of $500, most people chose the riskier option A [9]. As more value is lost from $0 to $500 than from $500 to $1000, the expected value of the outcome is less than the guaranteed loss of $500.

Another implication of prospect theory is that preferences will depend on how a problem is framed [8]. If the decision maker perceives the outcome as a gain they will accordingly act in a risk-adverse manner, while acting in a risk-taking manner where the perceived outcome is a loss.

Framing has been found to play an important role in medicine. How the benefits of a vaccine are framed will yield very different take-up rates [11]. Patients who are told a vaccine is guaranteed to stop a virus are more likely to take it, than patients who are told a vaccine will stop two equally likely viruses with 50% probability.

However, these results would suggest that most people would not buy insurance. Replicas of the original experiments into prospect theory have found that most people act differently depending on whether the cost is framed as a gamble or an insurance [20,21]. When a cost scenario is framed as insurance most people are risk adverse preferring to pay the smaller fixed amount, while when it is presented as a gamble people prefer to take on the risk.

Kahneman and Tversky [9] also found that people overweighed low probabilities and underweighted higher probabilities. This explains why people are prepared to buy lottery tickets.

2.3 Using Prospect Theory to Explain Requirements Selection Decisions

Given prospect theory can be applied to decision-making processes in disciplines from finance to medicine [22], this section proposes how prospect theory can be used to explain the requirements selection decisions in MDSPD – in this context losses are described in terms of costs and gains in terms of revenues.

The requirements types presented in Section 2.1 are represented in Table 3 with information on whether each is perceived as generating costs or revenues. Commercial requirement types are marked with an asterisk, while system requirement types are not.

Table 3. Requirement associations and related risk

Requirement Type	Cost/Revenue Driver
Customer specific features	* Revenue
Market pull requirements	* Revenue
Innovation	* Revenue
External Quality aspects	* Revenue
System Requirements (internal quality)	Cost

Commercial and system requirements are obviously very different in nature. Commercial requirements are clearly associated with customers – and therefore with revenue, while system requirements are associated with costs. This result can be seen through the language used to describe the different requirement types. Commercial requirements are referred to as *business opportunities*, and are discussed in terms revenue generation and expanding markets. In contrast system requirements are discussed as costs and must be justified with either, "If we do not do this now it will lead to higher costs in the future," or, "If we make this change now we will be able to lower our production costs."

As commercial and system requirements can be described in terms of gains and losses, prospect theory has an obvious connection in how these different requirement types are valued. This is seen most clearly by replacing gains and losses with revenue and costs, and placing requirements of different types on the graph in Figure 1.

Looking at a commercial requirement with an expected gain, one can see that a doubling of the gain does not double the value of the requirement. Similarly halving the gain does not halve the value of the requirement. The results from previous studies of prospect theory would therefore suggest that decision makers would be risk adverse when selecting commercial requirements, preferring guaranteed returns over larger riskier returns. Conversely decision makers would prefer to take more risk with system requirements, trying to minimize costs, even if these savings come at the risk of spending more money.

Different levels of risk are associated with the various requirement types. A customer specific requirement often has a customer willing to foot the entire cost, and one may reasonably assume that an individual customer is indicative of wider market demands. The return on investment of an innovation, however, does not have the same guaranteed revenue stream, but there is a potential for large revenues if the requirement proves popular. Internal system quality requirements are likewise risky as they involve investments in the longer-term sustainability of a software product. This could mean, for example, a requirement is developed allowing different database systems to be used with the product, but if this is never used the cost was unnecessarily incurred. It should also be noted that the customer is only aware of commercial requirements, with only the development organisation having clear visibility into internal system quality.

Given the clear link between prospect theory and MDSPD RE, the aim of the experiment presented in this paper is to investigate whether prospect theory holds in an MDSPD RE setting. While the authors acknowledge MDSPD RE is a complex decision making setting, the focus of this paper is on understanding the bias towards the perception of value as proposed by prospect theory, and how it may impact the decision making process.

3 Experiment Planning and Operation

This section presents the research question and major steps in the experiment planning and operation.

3.1 Research Question

The goal of the experiment presented in this paper is to investigate if the results of the experiments conducted by Tversky and Kahneman [9,10] can be seen in the MDSPD RE context - that is people are more risk taking when it comes to losses (costs) and risk adverse when it comes to gains (revenue).

In order to investigate this idea, the experiment follows the arrangement of the experiments conducted by Tversky and Kahneman [9,10], but places the participants in a MDSPD RE context. Thus the research question for the experiment is formulated as follows:

– When selecting requirements, do product managers show more risk-adverse behaviour with requirements formulated in terms of revenue, than with requirements formulated in terms of cost?

3.2 Experiment Instrumentation

A questionnaire was developed to investigate whether the participants behave in a manner that can be predicted by prospect theory. A range of risk levels and monetary values was used. The questions included are presented in Table 4.

Table 4. Question Sets for Revenue and Cost

Set	Revenue	Cost
1	Select one requirement: **A** has a guaranteed revenue of €10,000. **B** could raise €2,400 revenue with 75% probability, and €32,800 revenue with 25% probability.	Select one requirement: **A** has a cost of €10,000. **B** could cost €200 with 30% probability and €14,200 with 70% probability.
2	Select one requirement: **A** has a guaranteed revenue of €10,000. **B** could raise €2,000 or €18,000 revenue with equal probability.	Select one requirement: **A** has a cost of €10,000. **B** could cost €2,000 or €18,000 with equal probability.
3	Select one requirement: **A** has a guaranteed revenue of €10,000. **B** could raise €200 revenue with 30% probability, and €14,200 revenue with 70% probability.	Select one requirement: **A** has a cost of €10,000. **B** could cost €2,400 with 75% probability and €32,800 with 25% probability.

Table 5. Questions from Related Studies

Ref	Gain	Loss
[10]	Select one: – A sure gain of $250. – A 25% chance to gain $1000, and a 75% chance to gain nothing.	Select one: – A sure loss of $750. – A 75% chance to lose $1000, and a 25% chance to lose nothing.
[9]	Scenario: In addition to whatever you own, you have been given $1000. You are now asked to choose between the alternatives: – A 50% chance of gaining $1000. – A sure gain of $500.	Scenario: In addition to whatever you own, you have been given $2000. You are now asked to choose between alternatives: – A 50% chance of losing $1000. – A sure loss of $500.

In order to control the variable for value gains and losses are only described in monetary terms. While providing a richer context would better simulate reality, it is not easy to assess what value participants place on the different aspects that make up this context.

Each set of questions is designed in the similar style used in the original experiments by Tversky and Kahneman [9,10]. Examples of the original questions can be found in Table 5.

From the questions presented in Table 4 and Table 5 it is easy to see the similarities between formulation of the questions from the authors' experiment and the original experiment. However, the question formulation differed between the original and authors' experiment with the original experiment using zero gain and zero loss as an option in each case. The effect of this difference in the design of the questions is discussed in Section 4.

A number of measures were undertaken to reduce systematic biases in the results. The order effect was addressed by systematically varying the order of the questions in the study. Additionally questions were placed between questions on revenue and cost to distract the participant from their previous answers.

A pilot of the questionnaire was completed prior to conducting the experiment. Two lecturers in software engineering completed the questionnaire, with both feeling they had a clear understanding of what was required, and were able to make informed decisions.

3.3 Experiment Subjects

The experiment involved 71 student participants completing either their final year of undergraduate studies or a masters in Software Engineering at Blekinge Institute of Technology, Sweden. The group consisted mostly of international masters students, who came from Pakistan, India, China, Iran, Nepal, Bangladesh, Nigeria, Germany and Jordan; with four undergraduate students, all Swedish.

Forty-nine of the participants had prior experience of working in the software industry. The Swedish students had experience from software engineering projects in terms of project courses which are run in tight cooperation with industry and very much resemble real development in industry. At the time of the experiment most of the students had completed the courses in Software Requirements Engineering and Software Verification and Validation, thus the students were assumed to be familiar with requirements engineering concepts.

3.4 Experiment Operation

The experiment was conducted at Blekinge Institute of Technology in the spring of 2008 during a single two-hour session.

Prior to starting the experiment the participants were introduced to the role of a software product manager and presented with the key performance indicators (KPIs) used to assess a person in this position. The list of KPIs included responsibility over the selection of requirements in a release to maximize the profit (return on investment) generated by the sales of a product. The KPIs remained visible for the duration of the experiment and the participants were encouraged to act in accordance to them.

4 Results and Analysis

This section presents the results, with an analysis and discussion.

4.1 Presentation of Results

The results of the experiment are presented in Table 6. The table shows the percentage of participants that chose the safe and risky options for the cost and revenue related questions in each of the question sets presented in Section 3.2.

As shown in Table 6, most participants were risk adverse when answering questions related to revenue, with most selecting the safer option. For the revenue questions presented in *Question Set 1*, 69% chose a safe option, in *Question Set 2* a similar 68% chose the safe option, while in *Question Set 3* 56% chose the safe option.

The results for the cost related questions show only a small difference in preference between the safe and risky options in *Question Set 1* and *Question*

Table 6. Participant responses (percentages)

	Revenue		Cost	
Set	Safe	Risk	Safe	Risk
1	0.69	0.31	0.49	0.51
2	0.68	0.32	0.68	0.32
3	0.56	0.44	0.52	0.48

Set 3. However, in *Question Set 2* the participants were more risk adverse, which was the opposite of what was expected (68% safe).

Comparing the answers between the revenue and cost related questions, the results show increased risk taking attitude in cost related questions for *Question Set 1* (from 31% to 51%) and *Question Set 3* (from 44% to 48%). The attitude towards risk is unchanged for *Question Set 2*.

4.2 Analysis

The results for revenue related questions show a preference for avoiding risk in each and every case, demonstrating alignment with aligned with the original experiment conducted by Tversky and Kahneman [9]. While the results showed more risk-taking behavior in the questions related to cost, the results were not as strong as the original prospect theory experiments.

The level of risk avoidance is higher for the revenue questions in *Question Set 1* and *Question Set 3* than for the questions regarding cost in the same question sets. When it comes to cost related questions, two of the three tested cases (*Set 1* & *Set 3*) do not show a strong preference to take or avoid risk. These combined results, when considered relative to one another, indicate a more risk-adverse behaviour with requirements formulated in terms of revenue, than with requirements formulated in terms of cost.

The results for *Question Set 1* and *Question Set 3* show a clear change in attitudes towards risk between requirements termed as costs and requirements termed as revenue. While the result for *Question Set 2* did not show a strong difference, the majority of the cases support the application of prospect theory to software requirements selection.

4.3 Discussion

The results of the experiment indicate that decision makers will be more risk adverse when choosing between requirements formulated in terms of revenue, compared to when choosing between requirements formulated in terms of cost. This provides ground for concluding that prospect theory can be used to understand and explain decision making in MDSPD requirements selection situation.

The observed attitude towards risk taking for cost related questions was not as strong as expected, however, aspects of this may be explained by the experiment design. The participants of the experiment are students and do not have the same sense of ownership and responsibility towards the money of a real product manager. This could mean that they were not as sensitive to losing money. For example, looking at the cost related questions in *Question Set 2*, it is reasonable to assume that the students did not see the value of taking the risky option because it was not their own budget, own reputation or job position that was at stake. Another difference is that in original experiments in prospect theory, the subjects were offered to decrease the loss to zero, which may have motivated more to risk as other psychological factors are involved with zero [23].

The authors expect an experiment involving professional software product managers would be more aligned with the original experiments as people in

this role have a greater sense of ownership in the product for which they are responsible and will face a greater loss in reputation for failing to meet budget than the experiment participants. This assumption is supported by findings of the study on application of prospect theory on software project decisions, where project managers' decisions were found to be aligned with prospect theory [11].

4.4 Validity Threats

Internal and external validity threats are most important to address for experiments in software engineering field [24] and social sciences [25].

Internal validity is concerned with identifying whether the observed result of the experiment is influenced by other factors than the experiment treatment. For the experiment presented in this paper an ordering effect (learning) and experiment instrumentation (the way questions are formulated) are the most significant threats.

Treatment order effect would means the order the questions are presented will affect the participants' answers. To minimize this threat, the order of cost and revenue questions were systematically varied in the study.

To minimize the risk associated with the question formulations, a pilot of the experiment was conducted involving three participants. The intention of the pilot was to discover ambiguities and improve the formulation of the questions.

External validity is associated with the possibility to generalize the outcome of the experiment. While using students as experiment subjects usually puts a threat for generalizing the results [26], most of the students participating in the experiment have prior industrial experience and are trained in the requirements engineering field at a masters level. However, the participants may not have felt the responsibility for the success of the product in the same way that a product manager in industry would. As discussed earlier in Section 4.3 this provides ground for assuming that an experiment involving professionals will be more aligned with the original experiments in prospect theory.

MDSPD RE operates in a much more complex setting than that modelled in this experiment, potentially impacting the generalisability of the result. For example, while it can be argued that software requirement selection decisions are more commonly group decisions, and not up to individuals as modelled in this experiment. However, the application of the results in a group situation should still be possible as research has shown than individuals' attitudes to risk are translated into a groups attitudes to risk [27]. Similarly, while requirements are more complex than questions of cost and revenue, in order to control the participants perceived gains and losses and how these impact value, the problems were reduced to monetary terms.

The results presented in this paper are inline with other findings from industry, recognising the importance commercial requirements with a fixed return over more variable market opportunities [6]. Similarly, literature recognises the risk-taking attitude towards system requirements [4,5,6,7].

5 Conclusions and Future Work

The intention of the research presented in this paper is to investigate if the ideas behind prospect theory, one of the prominent theories in decision making, can explain requirements selection decisions in market-driven software product development (MDSPD). The experiment presented is this paper replicates the design of the original experiments into prospect theory, but places it in the context of requirements selection situations in MDSPD.

The experiment results show a clear potential for applying prospect theory in MDSPD setting. The participants consistently displayed risk adverse behaviour when selecting between requirements described in terms of revenue. In two of the three cases investigated the participants were more risk taking when selecting between requirements described in terms of cost when compared to the revenue situation. The increase in risk taking attitude was not as distinct as anticipated. However, the authors' expect that an experiment involving professionals would show larger difference between risk taking approach between the revenue and cost related situations.

To the best of the authors' knowledge the work presented in this paper is the very first application of prospect theory in MDSPD requirements selection context. The insights found in this paper provide contributions both to researchers and practitioners in this field, as they open up possibilities for explaining the behaviour of decision makers in different requirements selection situations. This provides an answer to why resent studies observe internal quality requirements being consistently undervalued compared to commercial requirements, as well as help in finding ways for managing the balance between different types of commercial requirements originating from market pull and technology push.

Looking more closely at commercial requirements, prospect theory suggests preference favour for the guaranteed revenue stream over more uncertain options. Looking at the different requirements types, this would translate to a preference for customer specific features over innovation related requirements, as predicting market demands brings with it more risk.

The situation for system requirements is more complex. As these requirements are viewed as cost-drivers, they cannot be directly compared with the other types of requirements that are perceived as delivering revenue. This highlights the need to describe system requirements in terms of the gains that they deliver, so that a comparison between requirements of different types can be made. However, it should be noted that the risk associated with system requirements is still higher – so even when described in terms of gains, natural preference will be given to less riskier options. Putting this in MDSPD context implies that unless there is a clear strategy for balancing commercial and internal quality requirements, the later will be consistently down- prioritized until the point when the architecture issues will pose an impediment for the production of commercial features.

A number of actions should be undertaken as future work. A follow-up study is planned involving software product managers from companies working in a market-driven software development context, helping address issues faced in the experiment presented in this paper. Additionally this work should be used to

educate software project managers to the biases they bring to the development process, and will be used as an input to a model to help software product managers balance requirement types when selecting requirements for a release.

References

1. Aurum, A., Wohlin, C.: The fundamental nature of requirements engineering activities as a decision-making process. Information and Software Technology 45(14), 945–954 (2003); Eighth International Workshop on Requirements Engineering: Foundation for Software Quality
2. Regnell, B., Brinkkemper, S.: Market-driven requirements engineering for software products. Engineering and Managing Software Requirements, 287–308 (2005)
3. Carlshamre, P.: Release planning in market-driven software product development: Provoking an understanding. Requirements Engineering 7(3), 139–151 (2002)
4. Wohlin, C., Aurum, A.: What is important when deciding to include a software requirement in a project or release? In: International Symposium on Empirical Software Engineering, pp. 246–255 (November 2005)
5. Wohlin, C., Aurum, A.: Criteria for selecting software requirements to create product value: An industrial empirical study. Value-Based Software Engineering, 179–200 (2006)
6. Barney, S., Aurum, A., Wohlin, C.: A product management challenge: Creating software product value through requirements selection. Journal of Systems Architecture 54(6), 576–593 (2008); Selection of best papers from the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2006)
7. Barney, S., Hu, G., Aurum, A., Wohlin, C.: Creating software product value in china. IEEE Software 26(4) (July– August 2009)
8. Plous, S.: The Psychology of Judgement and Decision Making. McGraw-Hill, New York (1993)
9. Kahneman, D., Tversky, A.: Prospect theory: An analysis of decision under risk. Econometrica 47(2), 263–291 (1979)
10. Tversky, A., Kahneman, D.: The framing of decisions and the psychology of choice. Science 211(4481), 453–458 (1981)
11. Lauer, T.W.: Software project managers' risk preferences. Journal of Information Technology 11(4), 287–295 (1996)
12. Karlsson, L., Regnell, B., Karlsson, J., Olsson, S.: Post-release analysis of requirements selection quality post-release analysis of requirements selection quality — an industrial case study. In: 9th International Workshop on Requirements Engineering — Foundation for Software Quality (RefsQ) (June 2003)
13. Regnell, B., Karlsson, L., Host, M.: An analytical model for requirements selection quality evaluation in product software development. In: Proceedings of the 11th IEEE International Requirements Engineering Conference, pp. 254–263 (September 2003)
14. Ngo-The, A., Ruhe, G.: A systematic approach for solving the wicked problem of software release planning. Soft Computing — A Fusion of Foundations, Methodologies and Applications 12(1), 95–108 (2008)
15. Karlsson, L., Dahlstedt, A.G.: Natt och Dag, J., Regnell, B., Person, A.: Challenges in market-driven requirements engineering — an industrial interview study. In: 8th International Workshop on Requirements Engineering — Foundation for Software Quality (RefsQ) (September 2002)

16. Gorschek, T., Wohlin, C.: Requirements abstraction model. Requirements Engineering 11(1), 79–101 (2006)
17. Keil, M., Mixon, R.: Understanding runaway it projects: Preliminary results from a program of research based on escalation theory. In: Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences, vol. 3, pp. 469–478 (January 1994)
18. Jorgensen, M., Carelius, G.J.: An empirical study of software project bidding. IEEE Transactions on Software Engineering 30(12), 953–969 (2004)
19. Chang, W.L., Yuan, S.T.: A markov-based collaborative pricing system for information goods bundling. Expert Systems with Applications 36(2), 1660–1674 (2009)
20. Hershey, J.C., Schoemaker, P.J.H.: Risk taking and problem context in the domain of losses: An expected utility analysis. Journal of Risk and Insurance 47(1), 111–132 (1980)
21. Slovic, P., Fischhoff, B., Lichtenstein, S.: Response mode, framing and information processing efforts in risk assessment. New directions for methodology of social and behavioral science. In: Question framing and response consistency, Jossey-Bass, San Francisco (1982)
22. Wu, G., Markle, A.B.: An empirical test of gain-loss separability in prospect theory. Management Science 54(7), 1322–1335 (2008)
23. Shampanier, K., Mazar, N., Ariely, D.: Zero as a special price: The true value of free products. Marketing Science 26(6), 742–757 (2007)
24. Wohlin, C., Höst, M., Runeson, P., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction. Springer, Heidelberg (2000)
25. Robson, C.: Real World Research: A Resource for Social Scientists and Practitioner-researchers. Blackwell Publishing, Malden (2002)
26. Berander, P.: Using students as subjects in requirements prioritization. In: International Symposium on Empirical Software Engineering (ISESE), pp. 167–176 (August 2004)
27. Farber, H.S., Katz, H.C.: Interest arbitration, outcomes, and the incentive to bargain. Industrial and Labor Relations Review 33(1), 55–63 (1979)

Toward a Service Management Quality Model

Gil Regev[1,2], Olivier Hayard[2], Donald C. Gause[3], and Alain Wegmann[1]

[1] Ecole Polytechnique Fédérale de Lausanne (EPFL),
School of Computer and Communication Sciences
CH-1015 Lausanne, Switzerland
{gil.regev,alain.wegmann}@epfl.ch
[2] Itecor, Av. Paul Cérésole 24, cp 568, CH-1800 Vevey 1, Switzerland
{o.hayard,g.regev}@itecor.com
[3] State University of New York at Binghamton and Savile Row, LLC, U.S.A
dgause@stny.rr.com

Abstract. [**Context and motivation**] Service Management has been steadily gaining in importance in many organizations and is becoming a major force of change in IT departments. ITIL, one of the main service management frameworks, defines the value of a service for its customers as the sum of the service utilities and service warranties but provides no specific rules for defining them. [**Question/problem**] Companies, IT departments and their consultants face difficulties defining utilities and warranties, as well as identifying their value for customers. [**Principal ideas/results**] We propose a general framework for understanding service requirements and for analyzing the quality of a service. The framework is based on General Systems Thinking. We define service utilities as norms created by the service for a given stakeholder. Service warranties then protect the stakeholder from variations of these norms as a result of threats. Value is created when the norms are maintained within the tolerance range of the stakeholder. Risk is defined as the possibility of detrimental consequences for a stakeholder if the norm is pushed outside its tolerance range. [**Contribution**] We believe that this work has the potential to advance theory and practice of service management in both academia and industry, and to reduce the risk of overlooking important service properties when defining service requirements.

Keywords: Service Utilities, Service Warranties, Norms, Tolerances.

1 Introduction

Service Management has been steadily gaining in importance in industry. As many IT departments are struggling with the changing business environment they are encouraged or sometimes forced to specify the services they provide to their customers, to define the value of these services, how they will be developed, provided, monitored and maintained.

Two of the main frameworks driving these changes are the IT Infrastructure Library (ITIL) and the Control Objectives for IT (COBIT). These frameworks have, in recent years, elevated the awareness of many organizations to the necessity of enhancing the

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 16–21, 2009.

strategic importance of IT departments for their parent organization by delivering services with customer value rather than applications and computing power.

The IT Infrastructure Library (ITIL) [9] is one of the main service oriented frameworks. It is a major industry driver that is drawing the attention of IT department managers to the service they provide and its value for their customers.

The concept of service[1] is defined in ITIL as [9] "a means of delivering value to customers by facilitating outcomes customers want to achieve without the ownership of specific costs and risks." Value is defined in ITIL as the sum of two deliverables, Utility and Warranty. Utility and warranty can be seen at a first glance as corresponding to Functional Requirements (FR) and Non-Functional Requirements (NFR), respectively, in Requirements Engineering (RE). Warranties are defined in Service Level Requirements (SLR), which, once signed by provider and beneficiary, become Service Level Agreements (SLA).

In ITIL Utility is defined as fitness for purpose whereas Warranty is defined as fitness for use. These two concepts are themselves defined as follows [9], "Fitness for purpose comes from the attributes of the service that have a positive effect on the performance of activities, objects, and tasks associated with desired outcomes. Removal or relaxation of constraints on performance is also perceived as a positive effect. Fitness for use comes from the positive effect being available when needed, in sufficient capacity or magnitude, and dependably in terms of continuity and security."

In everyday language [7] utility evokes both fitness for purpose and fitness for use, and relation between warranty and fitness for use is not directly apparent. The blurry nature of the definitions of ITIL results in confusion about the categorization of the properties of a service into utility or warranty. This confusion is one of the difficulties in specifying an SLR.

In this paper we attempt to clarify the crucial concepts of service level requirements, i.e. value, utility, warranty, and risk. We propose a conceptual framework based on General Systems Thinking (GST) [12], in which we consider a service as a system that maintains stable states (norms) for its stakeholders. We use the very simple example of an email service offered by an IT department to its company users. Due to space constraints, we only consider the availability warranty.

In Section 2 we present a very short introduction to GST. In Section 3 we discuss the view of a service as a system. In Section 4 we describe our conceptual framework. In Section 5 we review some of the related work before concluding in Section 6.

2 Viewing a Service as a System

One way of exploring the quality of a service is to model the service as a system, i.e. "a set of elements standing in interrelations" [11]. As demonstrated by Weinberg [12], some person, usually referred to as an observer in General Systems Thinking (GST), must define what the system is, its elements and their relationships, or else there is no system. A system is therefore a model (Weinberg calls it a viewpoint) that the

[1] We refer to business service in this paper as opposed to web services as they are considered in Service Oriented Architecture (SOA) and Service Oriented Computing (SOC).

observer creates in order to understand some object in his or her reality. From this philosophical point of view, the quality of a system is a relationship between an observer and the object itself. Quality is therefore neither an absolute property, nor an intrinsic property, of an object. It depends on the observer as much as it depends on the object.

In Systems Theory [11] a system draws energy, information and matter from the systems with which it has relationships in order to maintain its internal order. The concept of open systems implies that systems must accept input from other systems in order to survive. In doing so, a system becomes dependent on the stability of the input it receives from the other systems' output. In GST, outputs and inputs are traditionally specified in terms of states. More specifically, according to Klir [6], "the set of instantaneous values of all the quantities of the system (external as well as internal) is usually denoted as the *state* of the system. Similarly, the set of instantaneous values of all the internal quantities of the system will be called the *internal state* of the system." The system exists in states and the inputs may alter the system states. This causes the system to emit outputs. If we model the inputs, outputs, and system as another super system, then the inputs, system, and outputs can be described in terms of a more encompassing state space. We refer to the stable state of a system as its norm [10], whether it applies to its input, output or internal state.

In RE the observers of a system (or in our case of a service) are its stakeholders. A stakeholder of a service derives its norms from the stable input he or she receives from the service. This input is the service's output. The stakeholder then becomes sensitive to the variations in the service's output.

3 Defining Utilities, Warranties, Value, Risk and Quality

In the example of the email service, a user we call Alice provides input to the service in the form of requests to connect to the service, to send messages and to receive messages. The service outputs can be: connection confirmation, delivering sent messages to their destination and displaying received messages. As Alice takes the habit of using the service it becomes dependent on it for his or her everyday work. This everyday work is the output expected by the company from Alice. If the service's output is not stable, e.g. if Alice cannot connect to it of if the service doesn't display Alice's messages, or doesn't deliver the messages she sends, Alice will not be able to ensure the stability of her work.

From Alice's point of view, the utility of the email service is the outputs it maintains, connection, displaying received messages and delivering sent messages. Alice expects the service to deliver these outputs wherever and whenever she needs them. This can be anytime, anyplace (in the office, at home or during business travels) or restricted to certain hours in the office. The corresponding service warranty for Alice is that the service will be available when and where she needs it. Hence availability is not only a question of time (as described by ITIL, see definition of warranty in the introduction) but also a question of place.

The value for Alice is her ability to perform her daily work reliably. The risk for Alice is the probability that the email service, by not displaying or delivering messages will prevent her from performing her work. By accepting to use the email

service, Alice and her employer benefit from the advantage of electronic communication and shape their work accordingly but become potential victims if it is unreliable.

To understand the other stakeholders' viewpoints we use Gause and Lawrence's classification of stakeholders. Gause and Lawrence [3] propose to categorize users[2] (or rather stakeholders) as: clients, designers, direct users, secondary stakeholders, tertiary stakeholders, frivolous stakeholders. Clients are defined by Gause and Lawrence as [3], "responsible for economic matters, which include the ultimate financial success of the product." A similar definition can be found in ITIL [9]. Direct users are those who enter into contact with the service.

The advantage of this classification for a service is that it includes the designers as stakeholders. Extending this list from product stakeholders to service stakeholders, we have to include the service provider as well. The designers may or may not be part of the service provider. If the service provider is an IT department, the designers of the utilities are often the developers. The designers of the warranties and the service providers are the IT operations people. Secondary or tertiary stakeholders include regulators (market and legal), competitors, business process owners within organizations, as well as the partners of direct users and clients. In the case of the email service, it is probably on an off the shelf application designed by a software development firm. To turn it into a service, the application is hosted by the IT operations who must define its availability considering a set of threats that may limit this availability.

Gause and Lawrence provide a further subdivision of stakeholders. *Favored stakeholders* are those for which the service is designed and provided, in our example, Alice. *Disfavored stakeholders* are those for which the service is designed to create inconveniences and difficulties. Disfavored stakeholders are those who create the threats that push the norm outside of the tolerance range of favored stakeholders. Making the service impractical for them to use is one way of protecting the interests of favored stakeholders. Disfavored stakeholders include people who are not authorized to use the service and people who can create damage whether authorized to use the service or not. We can think of many such stakeholders for the email service, spammers, hackers, and even maintenance people can bring the service down. *Ignored stakeholders* are those for which the service is not designed at all, in our example, people outside the company with whom Alice never exchanges emails.

Favored stakeholders are those for which the service warranties maintain a norm within their tolerance range. Conversely, we want the warranties to be outside the tolerance range for disfavored stakeholders.

Based on this stakeholder classification, we propose the following definitions:

- A service utility is a norm that the service must maintain in order to satisfy favored stakeholders.
- The service warranty is the commitment by the service provider that the variation of the utility will be kept within favored stakeholders' tolerances and outside of disfavored stakeholders' tolerances.

[2] Gause and Lawrence's definition of a user as "any individual who is affected by or who may affect your product" corresponds to what is referred to as stakeholder in the current RE literature. We therefore take the liberty to refer to their concept of user as stakeholder.

- Value is the effect on a favored stakeholder when the utility is delivered within the warranty
- Risk is the possibility that a favored stakeholder will suffer the consequences of a service utility moving beyond their tolerance as a result of the system experiencing a given set of threats.

Favored stakeholders derive a value from the relationship they have with the service because it helps them to maintain their norms but at the same time they take the risk that if the service cannot maintain stability in its output, they may not be able to maintain stability in their own norms either. Hence, value and risk are inseparable.

In order to maintain the service's value to its favored stakeholders and to reduce the risk they are taking in using it, the service designer must design mechanisms that guarantee that the output remains within the tolerances of its favored stakeholders in the face of variations in the input. Limiting the variations of the input includes limitations to both favored stakeholders and disfavored stakeholders requests.

In our example, it is critical to understand Alice's tolerances for a lack of availability of the email service. These depend on the nature of Alice's work and on her personal preferences. If Alice depends on email for a mission critical work, her tolerances for a lack of availability will be very low. The service warranties will have to be very stringent. If, however, Alice uses her email for sending and receiving non urgent messages, the service can be down for maintenance every now and then and she may not even notice it. The IT department must therefore understand these norms and tolerances in order to define the SLR for the service provided to Alice.

When the variations in the utility become unacceptable to a stakeholder, he or she will define the service as being of poor quality. Conversely, when the states expected by a stakeholder are maintained despite perturbations, he or she is likely to declare that the service has high quality.

We can therefore define the quality of a service for a given stakeholder as the adequacy between its utilities and the needs of the stakeholder and adequacy of the warranties with the stakeholder's tolerances to variations in the utilities.

4 Related Work

Our definition of quality can be seen as an extension of the definition of service quality defined in [13], "the extent of discrepancy between customers' expectations or desires and their perceptions." Indeed, each stakeholder has his or her own idea of the quality of a service. We have also sharpened the question of expectations and perceptions.

Value-based Requirements Engineering [1, 5] is a stream of RE research based on the analysis of value exchanges within a network of actors. Value is considered in financial terms as the exchange of goods for money.

The research presented in [2] and [8] is an attempt to define business service properties by abstracting from the domain of software services. The result is much more technical and less general than our proposal.

The dichotomy between utilities and warranties in ITIL is similar to the well known dichotomy between Functional and Non Function Requirements in RE. Studies such as [4] can help in clarifying this dichotomy and therefore establish a better understanding of utilities and warranties.

5 Conclusions

Utilities and warranties as presented in ITIL are as important to service management as the concepts of Functional and Non Functional Requirements are to Requirements Engineering. We have found the definitions of utilities and warranties to be somewhat confusing. In this paper we propose to clarify these notions by resorting to the fundamental principles provided by General Systems Thinking, As a result we define utilities and warranties as relating to stakeholders' norms and tolerances. We believe that this clarification will be a stepping stone for further research in service science as well as a more pragmatic approach to service level requirements in industry. No doubt, more research is needed to refine the model we proposed. For example, we have not dealt with the issues of service support and service innovation. We have encouraging initial experience using this model in organizations, but more experience is needed.

References

1. Aurum, A., Wohlin, C.: A Value-Based Approach in Requirements Engineering: Explaining Some of the Fundamental Concepts. In: Sawyer, P., Paech, B., Heymans, P. (eds.) REFSQ 2007. LNCS, vol. 4542, pp. 109–115. Springer, Heidelberg (2007)
2. Ferrario, R., Guarino, N.: Towards an Ontological Foundation for Services Science. In: Domingue, J., Fensel, D., Traverso, P. (eds.) FIS 2008. LNCS, vol. 5468, pp. 152–169. Springer, Heidelberg (2008)
3. Gause, D.C., Lawrence, B.: User-Driven Design: Incorporating users into the requirements and design phase. Software Testing & Quality Engineering (January/February 1999)
4. Glinz, M.: On Non-Functional Requirements. In: Proc. 15th IEEE International Requirements Engineering Conference, New Delhi, India (October 2007)
5. Gordijn, J., Akkermans, J.M.: Value-based requirements engineering: exploring innovative e-commerce ideas. Requirement Engineering Journal, 114–134 (July 2003)
6. Klir, G.: An Approach to general Systems Theory. Van Nostrand Rheinhold, New York (1969)
7. Merriam-Webster online dictionary, `http://www.merriam-webster.com` (accessed, November 2008)
8. Nayak, N., Nigam, A., Sanz, J., Marston, D., Flaxer, D.: Concepts for Service-Oriented Business Thinking. In: Proc. IEEE International Conference On Services Computing, Chicago, Illinois, September 18–22 (2006)
9. Office of Government Commerce, ITIL Service Strategy, TSO, London (2007)
10. Regev, G., Wegmann, A.: Where do Goals Come From: the Underlying Principles of Goal-Oriented Requirements Engineering. In: Proc. 13th IEEE International Requirements Engineering Conference (RE 2005), Paris (September 2005)
11. von Bertalanffy, L.: General System Theory. George Braziller, New York (1968)
12. Weinberg, G.M.: An Introduction to General Systems Thinking. Wiley & Sons, New York (1975)
13. Zeithaml, V.A., Parasuraman, A., Berry, L.L.: Delivering Quality Service: Balancing Customer Perceptions and Expectations. Free Press, NY (1990)

A Controlled Experiment of a Method for Early Requirements Triage Utilizing Product Strategies

Mahvish Khurum[1], Tony Gorschek[1], Lefteris Angelis[2], and Robert Feldt[1]

[1] Blekinge Institute of Technology, Department of Systems and Software Engineering, S-372 25, Ronneby, Sweden
mkm@bth.se, tgo@bth.se, rfd@bth.se
[2] Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece
lef@csd.auth.gr

Abstract. [Context and motivation] In market-driven product development of software intensive products large numbers of requirements threaten to overload the development organization. It is critical for product management to select the requirements aligned with the overall business goals, product strategies and discard others as early as possible. Thus, there is a need for an effective and efficient method that deals with this challenge and supports product managers in the continuous effort of early requirements triage [1, 2] based on product strategies. This paper evaluates such a method – A Method for Early Requirements Triage Utilizing Product Strategies (MERTS), which is built based on the needs identified in literature and industry. **[Question/problem]** The research question answered in this paper is "If two groups of subjects have a product strategy, one group in NL format and one in MERTS format, will there be a difference between the two groups with regards to effectiveness and efficiency of requirements triage?" The effectiveness and efficiency of the MERTS were evaluated through controlled experiment in a lab environment with 50 software engineering graduate students as subjects. **[Principal ideas/results]** It was found through results that MERTS method is highly effective and efficient. **[Contribution]** The contribution of this paper is validation of effectiveness and efficiency of the product strategies created through MERTS method for requirements triage, prior to industry trials. A major limitation of the results is that the experiment was performed with the graduate students and not the product managers. However, the results showed that MERTS is ready for industry trials.

Keywords: Market driven requirements engineering, requirements triage, product strategies, MERTS, experiment, effectiveness and efficiency.

1 Introduction

Due to the emergence of markets for off-the-shelf or packaged software [3, 4], market-driven development is gaining increased interest in comparison to customer-specific system development [5, 6]. As a consequence a shift in focus is occurring, affecting software development in general and requirements engineering in particular [6]. In

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 22–36, 2009.

contrast to traditional requirements engineering, requirements in market-driven requirements engineering (MDRE) to a large extent come from internal sources such as developers, marketing, sales teams, support, bug reports, as well as from external sources such as different users, customers from different and multiple market segments, and competitors [7]. The result is a large and continuous flow of requirements that threaten to overload the development organization [5]. This has two major implications. One, the product and domain knowledge reside largely with the development company itself. For example a developer of robotics products with many of e.g. car manufacturers as customers probably knows more about robotics than any one customer.

Two, the risk and cost of development is carried by the development organization, meaning that the potential revenues depend on selecting the "right" requirements for implementation. The selection accuracy is the main success criteria for the development organization, and being able to perform the selection in a scalable and cost effective way is crucial to avoid overloading. Which requirements to select is a trade-off between different strategic factors such as key-customer requirements and long-term aspects and innovation efforts. All of these factors, and more, need to be explicitly stated and weighed together to reach an optimal strategy for the company, which can then be used for selecting the "right" requirements for implementation.

However, while industry managers regard strategy formulation and use as the most important aspect of technology management [8], strategy formulation is usually performed ad-hoc, and a systematic approach for formulating strategies is often missing in practice [9]. Even if the formulation of strategies was pursued, the factors affecting strategy formulation differ between different stakeholders. Strategic and middle management and technical experts all need to share one vision. Strategic managers often overlook the technical perspective, and technical experts can be unaware of or overlook the strategic managers' perspective. As a result of these challenges, identified both in academia and through industry case studies, a Method for Early Requirements Triage and Selection (MERTS) [10] was developed to combine both strategic and technical perspectives for the formulation of product strategies that are good-enough to be used for early requirements triage and selection.

This paper presents an experiment testing some key aspects of this method, following a stepwise plan to validate MERTS prior to industry piloting.

MERTS has two main purposes. First, it acts as a stepwise guide to creating product strategies taking both strategic and technical views into account thus following a systematic way of agreeing on a joint plan. Secondly, the strategies resulting from MERTS can be used by product managers to effectively perform requirements triage and requirements selection in a reasonable amount of time as spending initial 10 minutes on triage versus 10 hours is super critical for industry. The experiment aims at testing the second purpose of MERTS. Thus, the main purpose of the experiment is to assess the efficiency and effectiveness of requirements triage utilizing strategy formulated and formatted using MERTS prior to industry piloting. Thus, this experiment is considered as a lab validation following the research approach suggested by Gorschek et al. [11] aimed at producing useable and useful research results and successful technology transfer.

Before describing the experiment and experiment results an introduction to MERTS is given in Section 2. Section 3 details the experiment design. Section 4 lists

the validity threats. Section 5 contains preparation and execution details. Section 6 presents the results and analysis, and finally Section 7 presents the conclusions drawn and plans for further work.

2 MERTS Background

MERTS is centered on ensuring that the five strategic questions for a product are answered explicitly [10]. Fig. 1 gives an overview of MERTS and the three main parts of the method. The goal of MERTS is to offer a clear method detailing how to reach consensus and a homogenous understanding of a product strategy. The product managers using the method are required to follow these three parts. Each part has several steps (see Fig. 1).

Part One – Early Requirements Triage. This part provides steps to create an initial product strategy for use in requirements triage.

A. Specify. In order to explicitly state the goals and objectives of a product, it is important to specify the directions of movement for the product deduced from the organization's mission statement. Thus it is important to answer the three strategic questions ((1) Where we want to go?, (2) How to get there?, (3) What will be done?) for each product.

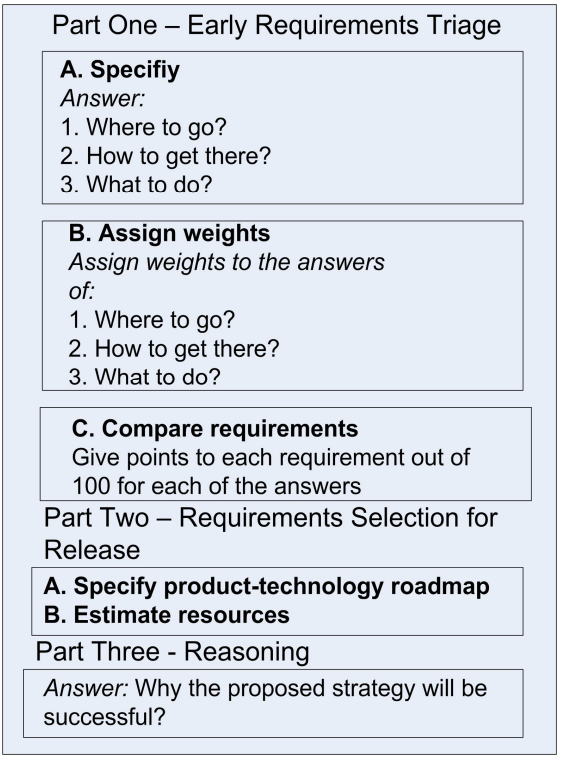


Fig. 1. MERTS Steps

The output of this step is an explicit understanding of goals and objectives associated with a specific product which can be used to perform requirements triage and selection for individual products.

To answer (A.1) "**Where to go**" the organization's directions of movement have to be clearly stated. An organization can have one or many directions of movement. For example, shareholders' revenue, profit, growth, and market share [10]. The answer to this question depends on identified directions of movement and their relative importance.

The answer to (A.2) "**How to get there**" will bind the strategy in terms of customer segments and competition targeted and differential advantage of the individual product providing a unique selling point. In order to answer this question there is a need to specify:

- Different customer segments targeted by a specific product, e.g. one customer segment can be the Asian market and another can be the European market. By explicitly specifying customer segments, relative priorities can also be assigned, helping in the selection of requirements. Customer segments can be defined either on a higher level of abstraction or refined depending on the needs of the organization.
- Competitors to a product to show which ones to target. This enables features provided by relevant competitors to be included in the product offering. Just as customer segments, competitors can also be prioritized relatively, giving more importance to features provided by high priority competitors.
- Differential advantage(s) of the product that makes it unique in the market place in relation to competitors. The differential advantage can be based on any one (or combination) of technology, pricing, strategic alliances and non-functional requirements. These can also be prioritized in relation to each other depending on their importance to offering the advantage. By identifying the differential advantages and prioritizing them, it is possible to ensure that all requirements are weighted against them and requirements providing unique differential advantages are not missed.

For the answer to (A.3) "**What to do**" a more management centered perspective can be used, focusing on product pricing, promotion, distribution, and service. However, since MERTS is targeted towards early requirements triage and selection, answers to this question will focus on the abstract technical considerations of a requirement. Some of the possible considerations rated highest by the technical experts during the interviews have been taken as example here, i.e. innovation, core assets, architecture stability, market-pull, technology-push, customization flexibility, and use of COTS [10]. Priorities can be assigned to each of these factors showing their relative importance with respect to each other.

B. Assign Weights. The answers from Step 1 are assigned weights. The rule is to assign weights to each of the factors based on their relative importance in a way that total weight remains 100. This way has been reported to be one of easiest and quickest prioritization methods [17].

C. Compare Requirements. The total weights of all the requirements are compared against a threshold to select or reject each of the requirements.

The first three steps of MERTS should be performed at product management level supporting the triage of requirements (aiding in the selection). The purpose of step 2 (Assign weights) is not requirements prioritization which is usually associated with

early project activities during release planning. The points assigned to each requirement, against each factor or sub-classification, show the level of strategic alignment.

Part Two – Requirements Selection for Release. After a set of requirements (deemed to be aligned with the strategy) have been selected, the question in focus is "when to get there". To answer this following two steps are required.

A. Specify product-technology roadmap. It has been emphasized in literature [12] to chalk out a product-technology roadmap to get an overview of the relationship between product releases (product evolvement) and successive technology generations. This means specifying what a product tends to achieve along the time axis in term of its evolvement and technology trends. This enables placement of requirements in appropriate intervals planned in a roadmap. For example, if a requirement requires expertise in some new technology to be explored in the future and this has been planned in the roadmap, the requirement can be postponed or depending on the urgency of the requirement, the roadmap can be altered.

B. Estimate resources. In order to determine the feasibility of the requirements, the organization needs to explicitly state financial and effort allowances against each interval in the roadmap. Several methods can be used to estimate cost, effort and time, e.g. feature points, function points, lines of code, and methods like e.g. COCOMO [13] can be used to support the efforts. An alternative could be to perform estimates based on previous development efforts. Additionally, requirements prioritization techniques [14] can be used to plan releases for the product.

Part Three – Strategy Rationale. Once the strategic questions have been answered, it is important to document the reasoning behind the decisions. This way if the decisions (and indirectly the answers) result in success (of a product) replication can be achieved, and the organization has good examples to follow for future efforts.

In addition, the strategy formulated through MERTS should be used to share product and organizational visions across the organization. In its simplest form it can mean writing a paragraph explaining the reason behind the answers, keeping in view the organization's long term goals, financial plans, technology trends and marketing trends.

In order to implement MERTS method, requirements need to be comparable to the strategies formulated. The reasoning is that MERTS is meant to assist in early requirements triage and selection. In case of requirements being too technical or too detailed method usage will not be efficient because it will be difficult to compare detailed technical requirements with strategies as strategies are formulated on a higher level of abstraction. Even if there is some process to compare detailed technical requirements with strategies they will still be too many detailed requirements to be compared against strategies. Often many detailed requirements form one product level feature/requirement therefore it is pointless to compare every detailed requirement against the strategies. Any method for abstracting the requirements can be used, e.g. the RAM model by Gorschek and Wohlin [6], as long as it produces requirements on an abstraction level comparable to product strategies.

3 Design of the Controlled Experiment

The usefulness and the usability of MERTS depend on several concepts that need to be assessed; the one studied in this controlled experiment is that it should be possible

to perform triage of a new incoming requirement based on its alignment with the MERTS strategy. This means that a MERTS strategy should be usable in an efficient and effective manner for performing requirements triage (formally stated as research questions in Section 3.3).

3.1 Context

The experiment was conducted in an academic setting, with the help of 50 engineering graduate students at Blekinge Institute of Technology. It was conducted as a mandatory although non-graded exercise at the end of a 7.5 ECTS merits master's course in research methodology. Participation was mandatory and despite the ethical issues of forcing subjects to participate in a study, it was believed that the experiment had several pedagogical benefits in the course. The students were instead given the option to exclude their individual results from the study, an option not utilized by any student. The intended users of MERTS, however, are product managers with several years of experience in a specific domain and product. In the experiment, the subjects have no training in using MERTS, they possess limited domain knowledge, are under time pressure, and most of them have not seen the product strategies or the requirements before. There is thus a considerable gap between the intended target group and the sample used in this experiment. The subjects in this experiment can be expected to adapt a more surface oriented approach to the problem than product managers. We argue that this works to our advantage, since any results that we evaluate are likely to stem from the instrumentation and the use of MERTS, rather than previous experiences of the subjects in the study. If MERTS proves to be usable in the experiment, it would indicate that it is able to decrease the dependency on individual's experience, product knowledge, and methodology.

3.2 Subjects

The group of experiment subjects using MERTS strategy for requirements triage had an average of 1.46 years of industrial experience, and only 3 out of 25 subjects had seen project strategies or performed requirements triage before. The subjects in the group using a natural language (NL) strategy for requirements triage had an average experience of 2.23 years, and 4 subjects out of 25 had seen product strategies in some form in their companies and 5 had performed requirements triage previously. This information was gathered through a post-experiment questionnaire; the groups were formed randomly.

3.3 Research Questions

The main research question is RQ which is described below along with associated hypotheses and independent/dependant variables.

RQ**:** If two groups of subjects have a product strategy, one group in NL format and one in MERTS format, will there be a difference between the two groups with regards to effectiveness and efficiency?

Hypotheses

Null hypothesis, $H_{o\ Effectiveness}$**:** The use of MERTS strategy for requirements triage is not significantly different from NL strategy with regards to effectiveness.

Alternative hypothesis, $H_{1\ Effectiveness}$: The use of MERTS strategy for requirements triage is significantly different from NL strategy with regards to effectiveness.

Null hypothesis, $H_{o\ Efficiency}$: The use of MERTS strategy for requirements triage is not significantly different from NL strategy with regards to efficiency.

Alternative hypothesis, $H_{1\ Efficiency}$: The use of MERTS strategy for requirements triage is significantly different from NL strategy with regards to efficiency.

Variables Selection: This experiment has the following independent variables:

Independent variables Product strategy formatted according to MERTS or according to NL.
Dependant variables The dependant variables are *effectiveness* and *efficiency* measured through: 1. Effectiveness: Number of correct requirements triage decisions. 2. Efficiency: Time taken (in minutes) to perform triage on all requirements.

The definition and hypotheses for finding an answer to RQ depict that the design is: *one factor with two treatments*. The factor is the product strategy and treatments are NL and MERTS.

3.4 Design and Instrumentation

Prior to the experiment execution one round of validation (pre-test) was performed to refine the experiment design and instrumentation. The pre-test was constructed and evaluated with the help of 4 colleagues at Blekinge Institute of Technology. Based on the experience from this pre-test, the experiment package was revised. Specifically, the initial presentation describing the study and running through an example was refined to make it more concrete for describing the motivation of the triage decisions taken by the subjects.

The subjects were divided randomly into two groups, with one treatment per group. The experiment consisted of two parts that both ran consecutively without breaks. The first part was a preparatory lecture where the concepts of requirements triage and MERTS/NL were introduced, together with a presentation of the experiment, research instruments and an example of how to take triage decisions and provide motivations. The second part of the experiment consisted of filling the forms. All other artifacts like the requirements and forms were the same.

During the experiment, the following instruments were used:

- Each subject was given an example of how to perform triage using either NL or MERTS (depending on the group).
- Each subject was given either NL or MERTS formatted strategy for the experiment. The product strategy detailed the goals of a new version of a mobile phone targeted for entertainment-oriented users in the Asian market. The level of information in the two strategies was the same with respect to goals and objectives, targeted customers and competitors, differential advantages and technical considerations. NL strategy was formulated based on example strategies given in literature. Industrial experience of authors with real product strategies was also beneficial to ensure that NL formatted strategy was as close as possible to industry practice. The MERTS strategy however, as

prescribed by the MERTS method, had weights assigned to each of the factors stated in the strategy which was absent in NL strategy because in traditional NL strategies the weights to each of the factors is not explicitly given in numbers rather stated as subjective statements.

- The requirements set contained 13 product and 18 feature level requirements. For example, messages communication, music playing, enhanced imagining, enhanced display, availability, usability, browsing, connectivity, and so on. The requirements were constructed to be of moderate to good quality based on industry observation. The appropriateness of the requirements and other instruments was also validated in the pre-test. It is important to understand that in lab experimentation, it is not possible to have a large number of requirements for triage. There is a limited amount of time where subjects have to understand the method and then apply it for requirements triage. The aspects of effectiveness and efficiency as evaluated in the experiment are however related to using MERTS strategies vs. NL strategies. The relative efficiency and effectiveness is the goal.

- Each requirement in the set has at least two levels: product and feature, and often also divided into functions. Each requirement was formatted and specified using the following attributes; Unique Id, Product level requirement, Feature level requirement, Function level requirement, Component level requirement (in some cases) and Comments.

- The instrumentation had a *Decision* column next to every feature level requirement with two options: Accept and Reject. For every triage decision the experiment subject had to specify a rationale behind the triage (Accept or Reject) decision. It was emphasized during the experiment training that the motivation had to be deduced from the product strategy and not personal judgments and opinions.

Last in the experiment, each subject had to answer the questions at the end of experiment as a post-test. The experiment materials (NL strategy, MERTS strategy, example requirements and the post-test) is not included in the paper as space does not allow, but can be obtained online at http://www.bth.se/tek/aps/mkm.nsf/pages/merts-experimentation.

4 Validity Evaluation

Internal validity. This threat can have a huge impact on the experiment results if the data collection forms and other instruments are poorly designed. To ensure that the research instruments, including the posed question, are of a good quality, one pre-test with the material was executed before the “live” round. Moreover, all subjects received the same introductory lecture, and were given the same material in the same order. It is thus unlikely that the instrumentation and the implementation of the treatment influenced the results unduly. That being said, since we used the answers of human subjects as measures, the gathered measures are of course not 100% repeatable.

To alleviate author’s bias towards MERTS while designing the experiment, a senior researcher (the second author) not involved in the creation of MERTS, was actively involved in the design of the experiment to avoid favoritism towards MERTS.

Construct validity. To reduce the risk of evaluation apprehension among the test subjects, they were told that they would be graded on their efforts, but not on the number of correct decisions.

External validity. To ensure the external validity and the ability to generalize the results, we use a requirements specification from a fairly mature domain. As discussed in Section 3.2, the knowledge and experience of the participants is less than that of the target audience (e.g. product managers in industry). To reduce this gap, a system from a domain that is familiar to the subjects was used.

The correlation analysis between the total number of correct triage decisions and the industrial experience show that there was no significant difference between performance of subjects with more industry experience and those with less experience (both for the group using MERTS strategy and group using NL strategy). Thus, the two groups were homogenous in terms of industry experience.

As the intended target of MERTS (e.g. product managers) would have not only a better requirement and domain understanding, but also more experience in triage, it can be argued that the ability to use MERTS (and the potential positive results obtained in the experiment) should be transferrable to industry practice. Moreover, experimentation using state-of-the-art research (well-structured method MERTS in this case) also has learning/training benefits for future professionals.

In this study paper printouts were used, which may impact the readability and the ease by which the participants may access the information. Hence, any positive effects are also here transferable to the target audience and the target environment as the use of tools may increase usability.

5 Operation

The subjects were not aware of the aspects intended for study, and were not given any information regarding research questions in advance. They were aware of the fact that it was a controlled experiment in the area of requirements engineering that was a part of their research methodology course. The experiment ran over a period of three hours, and the subjects were randomly divided into two groups seated in two different rooms. Introduction to the experiment was given during these three hours in the form of a brief slide show presentation. In this presentation basic concepts of product strategy and requirements triage were explained along with examples.

The mean time to conduct the experiment was around 60 minutes when using MERTS strategy, the shortest time spent was around 33 minutes and the longest was 107 minutes. The group using NL strategy had a mean time of around 33 minutes, the shortest time spent was 17 minutes and the longest was 50 minutes.

6 Results and Analysis

6.1 Testing $H_{o\ Effectiveness}$

In each group 18 feature level requirements were given to the subjects and they had to decide which of these are to be selected/rejected in accordance with the product strategy (either MERTS or NL). According to the experiment design 10 feature level

requirements were to be selected and 8 rejected based on the product strategies. During this analysis, answers that were in line with the study design and aptly motivated were treated as "correct". If an answer is in line with the study design but missing a proper motivation (that is the motivation is not based on the given product strategy) or if the answer is not in line with the study design, the answer is considered "incorrect".

Table 1 shows the mean, standard deviation, skewness and kurtosis for the total number of correct decisions for all the 18 feature level requirements for the two strategies: MERTS and NL respectively. The results show that the average number of correct decisions using MERTS (Mean = 17.72) is more than double the average number of correct decisions using the NL (Mean = 6.22).

Table 1. Statistics for total number of correct decisions for MERTS and NL strategies

MERTS		**Natural Language**	
Statistic	**Value**	**Statistic**	**Value**
Mean	17.72	Mean	6.22
Median	17.50	Median	5.00
Std. deviation	4.456	Std. deviation	4.124
Skewness	-.143	Skewness	1.180
Kurtosis	-1.257	Kurtosis	0.639

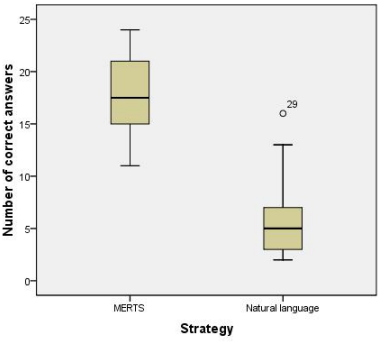


Fig. 2. Boxplots for total number of correct decisions using the two strategies

Confirmed complementary view is offered by the boxplots in Fig. 2 where the greater number of correct triage decisions using MERTS strategy is evident. Through the boxplots, an outlier (marked with a small circle and 29) was identified which is discussed below.

Table 2. Normality tests for total number of correct decisions

Strategy	**Kolmogorov-Smirnov**		**Shapiro-Wilk**	
	Statistic	Sig.	Statistic	Sig.
MERTS	0.158	0.200	*0.923*	0.146
NL	0.203	0.048	0.862	0.013

The skewness and kurtosis values for the total number of correct triage decisions show that the distributions seem to differ from the normal distribution. To check normalization prior to the application of an appropriate statistical test, normality tests were performed on the given data and the results are shown in Table 2. It can been in

Table 2 that the total number of correct triage decisions for MERTS do not differ significantly from the normal distribution (Significance = 0.20 > 0.05) but the distribution of the total number of correct triage decisions for NL is not normally distributed (Significance = 0.048 < 0.05). Based on this result the Mann-Whitney U test was used to compare if the two sample distributions (of total number of correct decisions using MERTS and NL strategies) come from the same population.

Looking at overall effectiveness of MERTS versus NL strategy the bar chart in Fig. 3 confirms that MERTS was more effective for triage decisions than NL.

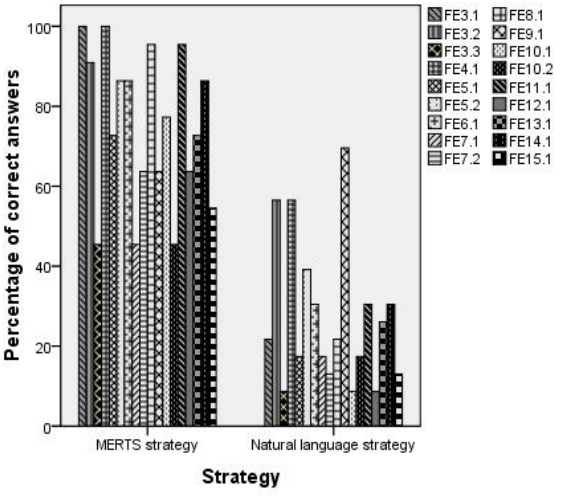


Fig. 3. Percentage of correct decisions in relation to strategy used

Additionally, the visual inspection of cross tabulations in Table 3 shows that the percentages of correct decisions for MERTS are significantly higher than the correct decisions for NL. For example, looking at second row it is possible to see that 22 subjects (62.9%) using MERTS strategy made a correct triage decisions for FE3.2 whereas only 13 subjects (37.1%) using NL strategy made a correct decisions. The difference of the percentages of correct and incorrect decisions between the two groups was tested with the chi-square test. For all the requirements the difference is significant at 0.025 except the requirement 9.1. This is the same requirement for which an outlier has been identified in Fig. 2. The reason behind this is that for this particular requirement the total number of correct decisions both for MERTS and NL are equal (16 correct decisions).

Requirement 9.1 is shown below with its related Function and Product level requirements. This requirement is easy to relate to in both strategy formulations, and also during the requirements engineering course at the university, students were given an example of a very similar requirement. In perfect hindsight it was not surprising that 50% subjects got this requirement decision correct both in MERTS strategy and NL strategy. The conclusion drawn after analysis was that the use of this particular requirement in the experiment was less than optimal.

Requirement 9.1.
Product: PR9: Usability Internationally **Feature: FE9.1: The mobile shall support multiple languages.** Function: FN9.1.1: The mobile shall provide Swedish language support FN9.1.2: The mobile shall provide Chinese language support FN9.1.3: The mobile shall provide Japanese language support.

Table 3. Significance using Chi-Square Test

Requirement	**Sig.**	**MERTS**				**Natural Language**			
		Incorrect		***Correct***		***Incorrect***		***Correct***	
		Count	**%**	**Count**	**%**	**Count**	**%**	**Count**	**%**
FE3.1	0.000	0	0	25	83.3	20	100	5	16.7
FE3.2	0.002	2	14.3	22	62.9	12	85.7	13	37.1
FE3.3	0.003	13	36.1	11	84.6	23	63.9	2	15.4
FE4.1	0.000	0	0	25	65.8	12	100	13	34.2
FE5.1	0.000	7	25	17	81	21	75	4	19
FE5.2	0.000	3	15.8	21	70	16	84.2	9	30
FE6.1	0.000	3	14.3	22	75.9	18	85.7	7	24.1
FE7.1	0.024	13	38.2	11	73.3	21	61.8	4	26.7
FE7.2	0.000	9	29	15	83.3	22	71	3	16.7
FE8.1	0.000	1	4.8	24	82.8	20	95.2	5	17.2
FE9.1	1.000	9	50	16	50	9	50	16	50
FE10.1	0.000	6	22.2	18	90	21	77.8	2	10
FE10.2	0.047	14	42.4	11	73.3	19	57.6	4	26.7
FE11.1	0.000	1	5.3	23	26.7	18	94.7	7	23.3
FE12.1	0.000	10	31.2	15	88.2	22	68.8	2	11.8
FE13.1	0.002	8	29.6	17	73.9	19	70.4	6	26.1
FE14.1	0.000	3	15	22	75.9	17	85	7	24.1
FE15.1	0.001	11	33.3	14	82.4	22	66.7	3	17.6

Finally to confirm the results, the Mann-Whitney U test is applied in order to check the significance of the results. Significance less than 0.001 was attained, indicating that there is a significant difference between the means of the two groups. The null hypothesis: $H_{oEffectiveness}$ is rejected and $H_{1Effectiveness}$ is confirmed, i.e. using MERTS is significantly different from NL for requirements triage with regards to effectiveness. To conclude, the use of MERTS strategy for requirements triage is superior to NL strategy with regards to effectiveness.

6.2 Testing $H_{0\ Efficiency}$

Fig. 4 shows the mean, standard deviation, skewness and kurtosis values for the time taken by the 50 subjects using the MERTS and NL strategies. The results show that average time taken using MERTS (Mean = 60.12) is double the average time taken using NL (Mean = 33.44). The outlier identified in Fig. 4 contributes to the large mean and standard deviation for the triage time taken using MERTS (Std. Deviation 19.10).

Table 4. Total time taken (minutes) for MERTS and NL strategies

MERTS		Natural Language	
Statistic	**Value**	**Statistic**	**Value**
Mean	60.12	Mean	33.44
Median	59.00	Median	34.00
Std. deviation	19.10	Std. deviation	9.10
Skewness	0.93	Skewness	-0.06
Kurtosis	0.62	Kurtosis	-0.92

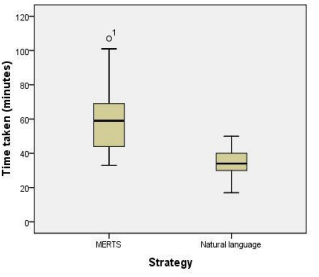


Fig. 4. Boxplots for time taken (minutes) for the two strategies

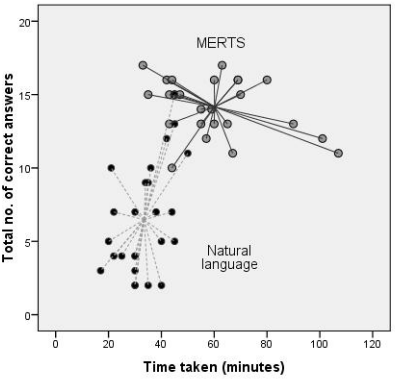


Fig. 5. Number of correct decisions versus time taken

Fig. 5 shows a scatter plot of the points representing each of the subject's responses in two dimensions (the dependant variables), the total number of correct decisions and the total time taken (in minutes). The spikes show the distance from the centroid. A clear difference between the two treatments can be seen. MERTS is characterized by long times and greater number of correct decisions whereas NL is characterized by shorter times and fewer correct decisions.

However, an analysis of the ratios of the total number of correct decisions in relation to total time taken using MERTS strategy (ratio value = 0.2355) and NL strategy (ratio value = 0.1937) shows that the time taken to perform correct triage decisions utilizing MERTS is only 0.12 times more than the time to perform correct triage decisions utilizing NL.

Using MERTS strategy, the number of correct decisions far outweigh the number of correct decisions using NL, thus it can safely be stated that MERTS has a fairly equivalent efficiency compared to NL, even if at a first glance MERTS may seem much more resource demanding.

Nevertheless, the subjects in the experiment that used MERTS did spend more time in total, and per correct decision, even if the latter was only marginal. A potential

explanation could be that the subjects using MERTS had to explicitly write a motivation/explanation referring to the strategy for every answer. This qualification of their decisions was not present on the NL side to the same extent as the NL strategy formatting was less exact the motivations were more of the character "could be good to have". The main motivation for demanding a thorough mapping between answers (choosing to accept or dismiss a requirement) and the MERTS formulated strategy was to enable decision traceability, a added value of MERTS that is not a part of the evaluation presented in this paper.

This might not explain the entire time difference, but at least parts of it. Using Mann-Whitney U, significance less than 0.001 was attained, indicating that there is a significant difference between the means of the two groups. This means that the null hypothesis $H_{o\ Efficiency}$ is rejected, and thus $H_{1\ Efficiency}$ is confirmed i.e. i.e. using MERTS is significantly different from NL for requirements triage with regards to efficiency. However, it cannot be concluded that use of MERTS for correct triage decisions is superior to the use of NL strategy with regards to efficiency. If the hypothesis was formulated as efficiency per correct answer, and if the time taken to write explicit qualification for the MERTS group was taken into consideration we feel confident that MERTS would be as efficient as NL, if not more.

7 Conclusions

MERTS is intended to aid product managers in performing requirements triage effectively and efficiently in a repeatable manner providing traceable decisions.

In this experiment the effectiveness and efficiency of requirements triage using MERTS was compared to using NL formulated strategies, which is the norm in industry. The main motivation of the experiment was to validate MERTS prior to industry piloting as such real industry tests require valuable and hard to obtain resources.

The experiment subjects were given 18 feature level requirements and asked to accomplish a considerable amount of work in a relatively short amount of time. The subjects were expected to form an understanding of the concept of product strategy and requirements triage, understand the domain (read and understand the requirements) and then take decisions whether to include or exclude a specific requirement based on the strategy supplied. The subjects were offered very little training, and they also possessed little prior knowledge regarding the domain compared to the level of a product manager in industry. Considering these aspects and the total number of correct decisions that resulted in using MERTS we feel that it is safe to draw the conclusion that MERTS is far superior to NL when it comes to strategy formulation and utilization for the purpose of requirements triage. The only potential drawback is that MERTS seems to be more resource intensive to use, although per correct answer we think that MERTS is at least as efficient as the NL option. Moreover, MERTS is essentially a systematic method for thinking and making decisions and that is why it takes more time but avoids errors. This systematic work is missing when using NL strategies.

The characteristics of industry are also relevant as real-life requirements triage utilizing product strategies would probably be easier for product managers than for the subjects in this controlled experiment. In industry, requirements triage and selection is not performed in isolation, regular meetings as well as official and unofficial conversations and discussions help in sharing views and reaching consensus. The benefit of MERTS is the ability to document the strategies (and the consensus) in a way that offers explicit decision support for all decision makers when performing requirements triage.

Considering these aspects, the results revealed through this experiment appear even more promising. In addition, product managers in industry are well versed in both their specific domain and in the field of requirements engineering. Given this, the use of MERTS would likely ensure even greater effectiveness and efficiency than was observed during the controlled experiment presented in this paper.

References

1. Davis, A.M.: The art of requirements triage. IEEE Computer 36, 42–49 (2003)
2. Simmons, E.: Requirements Triage: What Can We Learn from a "Medical" Approach? IEEE Software 21, 86–88 (2004)
3. Carmel, E., Becker, S.: A process model for packaged software development. IEEE Transactions on Engineering Management 42, 50–61 (1995)
4. El Emam, K., Madhavji, N.H.: A field study of requirements engineering practices in information systems development. In: Proceedings of the Second IEEE International Symposium on Requirements Engineering, pp. 68–80. IEEE Computer Society, Los Alamitos (1995)
5. Karlsson, L., Dahlstedt, Å., Nattoch Dag, J., Regnell, B., Persson, A.: Challenges in Market-Driven Requirements Engineering - an Industrial Interview Study. In: Proceedings of the Eighth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 2002), pp. 101–112. Universität Duisburg-Essen, Essen (2003)
6. Gorschek, T., Wohlin, C.: Requirements Abstraction Model. Requirements Engineering journal 11, 79–101 (2006)
7. Potts, C.: Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software. In: Proceedings of the Second IEEE International Symposium on Requirements Engineering, pp. 128–130. IEEE, Los Alamitos (1995)
8. Scott, G.M.: Top priority management concerns about new product development, vol. 13. The Academy of Management Executive (1999)
9. Krishnan, V., Karl, T.U.: Product Development Decisions: A Review of the Literature. Manage. Sci. 47, 1–21 (2001)
10. Khurum, M., Aslam, K., Gorschek, T.: MERTS – A method for early requirements triage and selection utilizing product strategies. In: APSEC 2007, Nagoya, Japan (2007)
11. Gorschek, T., Garre, P., Larsson, S., Wohlin, C.: A Model for Technology Transfer in Practice. IEEE Softw. 23, 88–95 (2006)
12. Kappel, T.A.: Perspectives on roadmaps: how organizations talk about the future. Journal of Product Innovation Management 18, 39–50 (2001)
13. Fenton, N.E., Pfleeger, S.L.: Software Metrics - A Rigorous & Practical Approach. International Thomson Computer Press (1996)
14. Berander, P.: Evolving Prioritization for Software Product Management. APS, PhD. Blekinge tekniska hogskola (2007)

Demystifying Release Definition: From Requirements Prioritization to Collaborative Value Quantification

Tom Tourwé, Wim Codenie, Nick Boucart, and Vladimir Blagojević

Sirris Software Engineering Cell
{tom.tourwe,wim.codenie,nick.boucart,
vladimir.blagojevic}@sirris.be

Abstract. [Context and motivation] Most software products are developed and improved over time in iterative releases. Defining the contents of the next product release is an important, but challenging activity, as a large number of potential requirements is typically available. **[Question/problem]** Implementing these requirements in a single release is impossible, and prioritizing them is hard: which requirements deliver the most value, and what is their value exactly? A study among European software companies in the context of the Flexi project revealed that this release definition challenge is still significant, in spite of the available state-of-the-art. **[Principle ideas/results]** This paper reports on a number of myths surrounding release definition we observed during the study, and explains shortcomings of the available state-of-the-art in a context where many requirements should be considered and defining and quantifying value is hard. **[Contribution]** We then propose a novel approach for reducing the risk of making wrong choices, based on emerging social technologies.

1 Introduction

Packaged software products are seldom fully finished in one shot, but rather follow an iterative development cycle [12]: different product increments are released over time consisting of a mixture of features, bug fixes, change requests and technical refactorings. Usually, software product builders are confronted with a large number of such requirements that can be included in the next increment [11]. Implementing them all at once is impossible, because of limited available resources, time restrictions or conflicts in the wishes formulated by the stakeholders. Choices have to be made and priorities have to be set [14].

In the context of the European Flexi research project, we are investigating this subject and have performed a study among more than 100 (Belgian) product builders. This revealed that release definition —the selection of the most valuable requirements to be implemented in the next product increment — fulfils an important strategic role. Companies realise that making incorrect choices for a release can significantly impact their competitiveness.

Given this strategic relevance, it is striking to observe that the studied companies often implement fragmentary and inconsistent solutions for defining releases, leading to suboptimal and unsatisfactory results. Their often ad-hoc approach can be explained by the fact that, despite its strategic relevance, release definition is not yet

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 37–44, 2009.

considered as an explicit and standalone software engineering discipline. Rather, it is scattered throughout many different disciplines, such as requirements engineering [20], release management or development methodologies such as Scrum [18].

In this paper, we first provide insights into how the studied companies approach the release definition challenge. Because of the fragmentary and scattered nature of the discipline, and because of the myriad of state-of-the-art tools and techniques available, we observed a number of release definition myths, which we discuss next. We then show that the available state-of-the-art is not optimal when the number of requirements to consider is large and defining and estimating the value of those requirements is hard. Finally, we present a novel approach that aims to reduce the risk of making the wrong choices for a release.

2 The Industry Perspective: Myths of Release Definition

Myth 1: Release definition boils down to installing an idea database

Many companies feel that they don't have a grip on the release definition challenge because they don't know all the ideas and requirements that are available for inclusion in the next product increment. In order to get an overview of those, they install a centrally-accessible idea database that makes it easier to search for ideas and to submit new ones. During our study, we have observed databases in different forms and shapes, ranging from excel files over issue tracking systems to dedicated idea management tools [8][9]. Some companies even open these up to their customers.

Although an idea database is necessary, it is not sufficient by itself for dealing with the release definition challenge. An idea database is in essence a feed of ideas that have no explicit value associated to them. In other words, the idea database itself does not help in selecting requirements. On the contrary, the success of an idea database can make the challenge worse, because many more ideas will have to be considered, ideas which are in different states of maturity, and at different levels of detail. Managing this amalgam of ideas quickly becomes a burden that can hinder the further success of the idea database.

Myth 2: Release definition boils down to appointing an enlightened product manager

Many organizations appoint a gatekeeper, in the form of the product manager, to resolve the release definition challenge. The product manager guards and structures the potential requirements and selects the best ones. He is expected to study the market, listen to customers and prospects, identify interesting opportunities, study the competition, their offerings and activities, etc. Based on this knowledge he is expected to separate the valuable requirements from the less valuable ones.

Given the complexity of today's products and markets, determining the value of a requirement resembles solving a puzzle; it requires different pieces of information in order to position the requirement in the larger context. These pieces of information are often dispersed among many different people in the organization; they are often incomplete and sometimes even tainted by personal agendas. Therefore in practice, the theory of the enlightened product manager does not stand. The product manager

should be supported in exploiting the knowledge that lives across the organisation to help him make the right choices.

Myth 3: Release definition can be expressed solely as a tayloristic process

Many companies look at the release definition problem purely from a process perspective. They chop up the release definition process in a series of well-defined phases, steps and activities, each requiring specific skills and associated responsibilities. Since these skills can be trained, the overall process is independent of the particular people participating in it. The end result of the process is thus in theory always the same.

These Tayloristic principles[1] work well in an environment where the activities and associated skills and responsibilities can be identified easily, where the same results need to be produced over and over again, where uncertainty is reduced to a minimum and where creativity is not required. Unfortunately, we observed during our study that the typical release definition environment holds none of these properties. A clear definition of the activities and skills related to release definition is still lacking. This classifies the problem as a wicked problem [2]. Additionally, defining the tasks and responsibilities of people up front, thereby restricting their input to particular information only, obstructs creativity and disregards the inherent uncertainty of software product development. Relying only on a process-oriented view clearly will not suffice to solve the release definition challenge.

Myth 4: Release definition boils down to using a voting system

An alternative way of inviting many different people to provide their input and views is involving them through a voting system. People are presented a list of requirements and are asked to cast their vote on the requirements they prefer. As different voting systems exist [22], such preference can be expressed in different ways: as a simple yes/no vote, for example, or as a number on a particular scale (e.g. 1 to 5). Requirements that get the most votes are assumed to be those that are the most valuable.

Although voting systems are simple and easy to organize, they do have some particular quirks. Casting a sincere vote for the best requirements requires that each and every requirement is considered, and that the different advantages and disadvantages of realizing these requirements are carefully balanced. This becomes time consuming if the list of requirements is large.

Voting also requires knowledgeable people that understand each and every requirement in the list in detail. Not surprisingly, this assumption does not hold. A sales person might not understand that a requirement that will generate a lot of revenue is too costly or too complex to implement, or a developer might not know that a costly requirement will generate a lot of revenue.

People also employ clever strategy and tactics, all because they want to avoid an unwanted result, rather than achieve the best result. Such tactical voting is unavoidable, whatever voting system is selected, and beats the assumption that stakeholders always handle in the company's best interest.

[1] This view originates from Frederic Winslow Taylor [21].

A last challenge associated with voting systems is that one looses track of the underlying rationale the voters used to cast their votes. Suppose two requirements get the same amount of votes: one requirement doubles the market share at a huge development cost, while the other one halves the maintenance cost but does not attract a single customer. Which of these two requirements is the most valuable and should be realized?

Myth 5: Release definition boils down to a requirements prioritization algorithm

People have looked for ways to overcome the disadvantages of voting while at the same time retaining its advantages. In other words, how to determine the real value from the overwhelming amount of potential requirements in a collective, time-efficient and objective fashion? This gave rise to extensive research on automated requirements prioritization through algorithms [13][23]. Requirements prioritization algorithms capture the value of a requirement in a single number, computed by means of a (complex) formula that combines the (values of) relevant criteria in a clever way. Based on this number, the list of requirements is sorted.

The concept itself is quite clear and promising, but living up to the promise in practice seems hard. Companies often do not know exactly what the relevant criteria are for expressing value in their context. Some approaches pre-define criteria such as cost, effort and risk [23], while others allow user-defined criteria, but provide no guidance on determining the criteria that are useful in a particular context [15].

Whether using pre-defined or user-defined criteria, we also heard questions about how to quantitatively measure them. Criteria such as cost and effort can be expressed in absolute terms (money or time spent), but some criteria, such as innovativeness or product stability, are much more difficult to express as numbers. In addition, only estimates can be provided and correct estimates are crucial for having a correct prioritization. Estimation is an art by itself, however.

3 The Research Perspective: Available State-of-the-Art

In trying to help companies solve the release definition challenge, we looked at the current state-of-the-art, and distilled 6 groups in which most of these available methods, tools and techniques can be classified. A recent overview of the available state-of-the-art in the domain of release definition is presented in [12][13].

- **Clustering in importance groups:** approaches that divide the requirements into a small number of different importance groups: Kano analysis [10], the MoSCoW method [3] and the Planning Game [1], for example.
- **Consensus-based approaches:** approaches specifically geared towards reaching consensus among a group of stakeholders. An example is the Delphi method [16].
- **Multi-criteria ranking:** approaches that automatically rank the requirements, based on the value of multiple relevant criteria and a specific formula that combines these values into a single value. Examples are Wieger's method [23] and Release Planner [15].

- **Pairwise comparison:** approaches that rely on mutually comparing all requirements, and identifying for each comparison the most valuable requirement. Examples are manual pairwise comparison approaches, such as the analytic hierarchy process [17] and the "20/20 vision" innovation game [7], and semi-automatic approaches, such as the one implemented by the FocalPoint tool [5].
- **Voting systems:** approaches that involve different stakeholders and ask each one of them to express their preference in some way or another. Examples are the "Buy a Feature" innovation game [7] and the "$100-test" approach.
- **Financial approaches:** approaches based on financial measures, such as the Internal Rate of Return, Net Present Value [4], or business cases.

Keeping in mind the myths, we studied how a company's context impacts the suitability of a particular release definition approach from these 6 groups. While there are potentially many relevant context characteristics, we have so far identified the following two: the number of potential requirements, and the ability of the company to quantify the value of those requirements.

The number of requirements available for the next product increment influences the trade-off between a lightweight, coarse-grained approach or a fine-grained, more heavyweight approach. The latter approaches can be used when few requirements are available and an in-depth analysis of each and every requirement is feasible. These approaches do not scale well when the number of potential requirements is high, however, as specifying the value of thousands of requirements in detail is too time-consuming. In such cases, coarse-grained approaches can be applied first, to make an initial selection, followed by ever more finer-grained approaches that complement the existing information, until making a motivated selection is deemed possible.

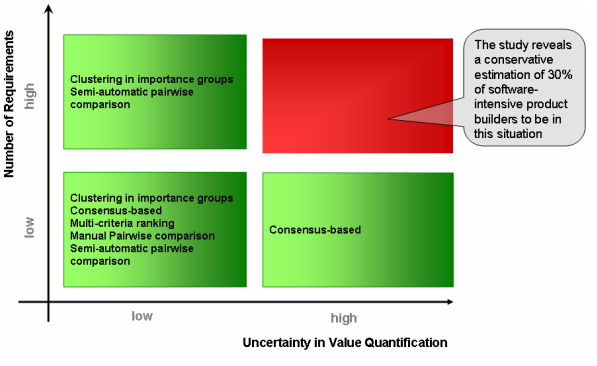


Fig. 1. Applicability of the state-of-the-art w.r.t. the number of requirements and the uncertainty in value quantification

The ability to quantify the value of requirements refers to how easily a company can express what "value" means in their context. When a company is building custom software for a specific customer, their definition of "value" will be different than that of a company that is building an off-the-shelf software product. In the first case, the customer might be available to estimate the value of the requirements and to comment on the next product increment. In the latter case, different customers may have

different wishes and opinions, and it is in general more difficult to predict how well a particular product increment will be received by the market.

Figure 1 illustrates how the 6 groups of approaches are positioned against the above characteristics. All approaches work fine when both the number of requirements and the uncertainty in value quantification is low. Few approaches however work well when either the number of requirements is high or the uncertainty in value quantification is high. Most striking however is that none of the approaches seems suitable in a situation where both the number of requirements and the uncertainty on the value quantification is high. This is exactly the situation that a lot of software intensive product builders face, according to our study.

4 Releasious: Exploiting the Wisdom of Crowds

An interesting question is what the core characteristics are of release definition approaches that can deal at the same time with large amounts of potential requirements and with high uncertainty in the value quantification. We feel a paradigm shift is ongoing: whereas in the past, release definition was often equated with requirements prioritization, it is becoming more and more a "social" interaction in which value is quantified collaboratively by all involved stakeholders.

Current evolutions in Web 2.0 social technologies are now sufficiently mature to allow exploitation in this challenging domain. In particular the application of wisdom of crowds techniques [19] seems a promising research path to consider: How can techniques such as crowd sourcing[2] and collaborative filtering [6] be applied to the problem of release definition and what benefits can they bring to companies?

To understand this, we are developing a web 2.0 prototype tool called Releasious™. Releasious allows organisations to

- **tap into their collective creativity potential**, by providing a centrally accessible platform that allows any stakeholder to submit his or her product ideas, comment on other ideas and/or state his opinion on their value.
- **determine the value of requirements**, by involving all stakeholders, allowing them to express their individual opinion on the value of a requirement, as well as providing information that they think is relevant. Stakeholders can use their own vocabulary when expressing the value of a requirement, by using free-form "tags" (e.g. "innovative", "sellable", "quality improving", …). Inspired by the work on folksonomies, this can allow to grow a company-specific definition of how value should be expressed.
- **let the controversial requirements emerge**, by using dashboard technologies that consolidate all the information available into appropriate views and signal disagreement. Word clouds are used, for example, to show what tags are considered most relevant for a requirement, and stakeholder agreement is measured by considering the distribution of different tags over different stakeholders.

[2] Crowdsourcing is the act of taking a task traditionally performed by an employee or contractor, and outsourcing it to an undefined, generally large group of people, in the form of an open call.

- **define product releases that are in lign with the product roadmap and the business strategy**, by supporting product managers in browsing through the available information by offering pre-defined and user-definable views and filters that help in making a motivated and rational selection of requirements.

We expect three major benefits from using such wisdom of crowds based techniques. A first one is based on a "divide and conquer" philosophy. The large and complex problem of release definition can be partitioned in smaller pieces, each being attacked by different (groups of) stakeholders. Afterwards, the results can be aggregated. This can allow a company to deal with a large number of requirements in a collaborative way. A second benefit is that through the collaborative assessment of value, the company-wide vision of what value is can be established and spread among the different stakeholders. This common understanding of value, in combination with aggregation and negotiation of individual value assessments coming from different stakeholders, will yield a third benefit: limiting the risk of wrong value assessment of individual requirements.

Several important challenges remain to be addressed, such as stakeholder motivation, reputation management and rewarding strategies. Furthermore, meeting the necessary conditions for effective wisdom of crowds application is also challenging [19].

5 Summary

In this article, we presented a number of release definition myths observed at various software builders. We found that current state of the art cannot be efficiently and/or effectively applied in their specific context. This brought us to propose a novel approach, based upon wisdom of crowds techniques. This new approach of course faces its own challenges, which are currently being studied with the help of a prototype tool called Releasious™. Experiments with Releasious™ are underway, but the results are still premature. Releasious™ can be seen in action at http://releasious.sirris.be.

References

1. Beck, K., Fowler, M.: Planning Extreme Programming. Addison-Wesley, Reading (2001)
2. Carlshamre, P.: Release Planning in Market-driven Software Product Development - Provoking an Understanding. Requirements Engineering Journal 7(3), 139–151 (2002)
3. Clegg, D., Barker, R.: Case Method Fast-Track: A RAD Approach. Addison-Wesley, Reading (2004)
4. Cohn, M.: Agile Estimating and Planning. Prentice-Hall, Englewood Cliffs (2005)
5. FocalPoint, `http://www.telelogic.com/corp/products/focalpoint/index.cfm`
6. Goldberg, D., et al.: Using collaborative filtering to weave an information tapestry. Communications of the ACM 35(12), 61–70 (1992)
7. Hohmann, L.: Innovation Games: Creating Breakthrough Products Through Collaborative Play. Addison-Wesley, Reading (2006)
8. IdeaScale, `http://www.ideascale.com`
9. InnoBar, `http://www.innobar.org`

10. Kano, N.: Attractive quality and must-be quality. The Journal of the Japanese Society for Quality Control, 39–48 (1984)
11. Karlsson, L., Dahlstedt, S.G., Regnell, B., Nattoch Dag, J., Persson, A.: Requirements engineering challenges in market-driven software development - An interview study with practitioners. Inf. Softw. Technol. 49(6), 588–604 (2007)
12. Karlsson, L.: Requirements Prioritisation and Retrospective Analysis for Release Planning Process Improvement, PhD Thesis, HUT / Department of Computer Science (2006)
13. Lehtola, L.: Providing value by prioritizing requirements throughout product development: State of practice and suitability of prioritization methods. Licentiate Thesis, HUT / Department of Computer Science (2006)
14. Lubars, M., Potts, C., Richter, C.: A Review of the State of the Practice in Requirements Modeling. In: Proceedings of the IEEE International Symposium on Requirements Engineering, pp. 2–14. IEEE Computer Society Press, Los Alamitos (1993)
15. ReleasePlanner, http://releaseplanner.com
16. Rowe, G., Wright, G.: The Delphi technique as a forecasting tool: issues and analysis. International Journal of Forecasting 15(4) (October 1999)
17. Saaty, T.L.: Fundamentals of Decision Making and Priority Theory. RWS Publications, Pittsburgh (2001)
18. Schwaber, K.: Agile Project Management with Scrum. Microsoft Press (2004) ISBN 0-7356-1993-X
19. Surowiecki, J.: The Wisdom of crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations Little, Brown (2004) ISBN 0-316-86173-1
20. The Software Engineering Body of Knowledge (SWEBOK), http://www.swebok.org
21. Taylor, F.W.: The Principles of Scientific Management. Harper & Brothers. Free book hosted online by Eldritch Press (1911)
22. Voting systems, http://en.wikipedia.org/wiki/Voting_system
23. Wiegers, K.: First things first. Softw. Dev. 7(9), 48–53 (1999)

Specifying Changes Only – A Case Study on Delta Requirements

Andrea Herrmann[1,*], Armin Wallnöfer[2], and Barbara Paech[3]

[1] Fraunhofer IESE, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
Andrea.Herrmann@iese.fraunhofer.de
[2] ABB Corporate Research Center Germany, Wallstadter Str. 59,
68526 Ladenburg, Germany
armin.wallnoefer@de.abb.com
[3] University of Heidelberg, Im Neuenheimer Feld 326, 69120 Heidelberg, Germany
paech@informatik.uni-heidelberg.de

Abstract. [Context and motivation] Requirements engineering methods and examples presented in textbooks and scientific publications usually treat software which is developed - and therefore specified - from scratch. However, in the software development practice, this situation is very rare. In an industry case study, we encountered the situation that a software system in use had to be enhanced by a small delta. **[Question/problem]** Our objective was to specify these delta requirements without having to describe the complete system in detail. Therefore we explored how much of the existing system had to be specified in order to make the delta requirements understandable. **[Principal ideas/results]** We made an intensive literature search to proven practices. As we were not successful we applied the requirements engineering method TORE and extended it to capture the delta requirements. **[Contribution]** In this paper we describe a process for capturing delta requirements. To our knowledge, this is the first work about this practically relevant question. In our case study, hierarchical refinement of requirements top-down and iterative requirements prioritization successfully supported the specification of deltas, combined with a high-level specification of the existing system. We also present our experiences during the case study and propose ideas for further research.

Keywords: change request; evolution; requirements specification.

1 Introduction

In practice, most often software projects enhance an existing system, e.g. when a software product is customized, when a legacy system is adapted to changed requirements or slightly enhanced, or when standard components are added. Nevertheless, methods and examples presented in requirements engineering (RE) textbooks and scientific publications, usually treat software which is developed and specified from scratch.

[*] The presented work is the result of her previous work at the University of Heidelberg.

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 45–58, 2009.

In the following, we call the requirements describing the enhancement "*delta requirements*". Such delta requirements could be specified as isolated units or based on and linked to a specification of the existing system. Such a context description supports the understanding and discussion of the delta´s influence on the existing system, especially when the stakeholders of the project do not know the existing system. However, in practice often a complete and up-to-date specification of the system does not exist. Thus, the existing system has to be specified before the delta requirements can be described. This is typically too time-consuming, especially when the system is large and the delta is small. Although a documentation of the complete system has its value in terms of knowledge management, the project budget does often not allow its creation.

In an industry case study, we encountered the situation that a software system in use had to be enhanced by including change management functionalities. Our objective was to focus the specification effort on these delta requirements mainly and to analyze the influence of the new functionality on the currently supported processes, without having to describe the complete existing system equally detailed. The approach we used was to apply the requirements specification method TORE (which stands for "Task and Object Oriented Requirements Engineering" [1], [2]) and to find out how much of the existing system has to be specified in order to make the delta requirements understandable. The value of TORE for this task lies in the fact that requirements are described on different levels of granularity. Refining requirements hierarchically should support describing the delta requirements in detail and the rest of the system on a higher level of granularity.

The case study described in this experience paper took place at ABB, a company which is a leader on the market on energy and automation technology and which employs 115.000 persons worldwide. ABB has adapted a systems engineering tool to their own needs and enhanced it. The resulting plant engineering tool - which we here call the Integrated Engineering System (IES) - is used internally in the field of power plant engineering. It integrates all project specific data – from project planning and RE to plant operation. Thus, it offers a consistent model of the plant and the project documents and links all project stakeholders within ABB automation projects: subcontractors as well as the customers. Change management is important in plant engineering projects because engineering results from the different stakeholders add up to a huge amount of data, which are hierarchically, logically and technically interdependent. This complexity makes such projects a very dynamic and sophisticated undertaking, where changes frequently take place and have far-reaching consequences. Such changes are caused for example by refinement of high-level requirements or by new technical or other constraints. It is essential to analyze impacts of changes before deciding about their implementation, to document and to communicate these changes. These activities serve risk management.

In the case study, it was investigated how change management functionalities can be added to IES. A prototype of the highest ranked requirements was implemented. The RE activities of the case study were managed by one of the authors who discussed the approach with the other two authors. The stakeholders who defined and prioritized the requirements were experts of ABB, who already use the IES system and are technical or business experts for plant engineering.

The remainder of this paper is structured as follows: Section 2 presents the result of our literature research. Section 3 describes how we proceeded in the case study, while Section 4 discusses our approach, especially how we modified TORE and why. Section 5 presents lessons learned. Section 6 is the conclusion of this publication and presents questions which we plan to investigate in future work.

2 Related Work

We searched literature in order to find answers to the following **questions**:

- Are there empirical studies which result in statistics about how often in software projects it is necessary to describe delta requirements?
- Which specific challenges are met in practice with respect to delta requirement specification in the situation that no complete specification of the existing system is available?
- Which solutions are proposed and/ or applied for the specification of delta requirements?

Typically, in the context of requirements evolution and maintenance, it is assumed that a specification of the existing system is given using the traditional specification methods. Then, the following topics are treated in the typical publications about delta requirements:

- Why and when requirements change and which software development process or tool can cope how with requirements evolution.
- How requirements changes (to the specification of the existing system) are managed and documented, e.g. by requirements management, traceability and rationale.
- How requirements can be specified in order to be reusable.

Thus, we made a more comprehensive literature search for approaches dealing specifically with delta requirements. As the term “delta requirements” is not used in the scientific literature so far, we successively used the following search keys: delta requirements; scalability AND requirements AND specification; scalability AND requirements AND modeling; scalable AND change request AND specification; change AND specification; requirements creep; requirements AND hierarchical AND refinement; incremental AND requirements AND specification; incremental AND requirements AND model; requirements AND specification AND evolution; COTS AND specification; COTS AND specification AND development; COTS AND requirements; COTS AND requirements AND development; “requirements change”; “requirements reuse”; Software AND maintenance AND requirement; “requirements recycling”; “perfective maintenance”; perfective AND maintenance AND requirements; perfective AND maintenance AND specification; change AND impact AND analysis; requirements AND change AND control; requirements AND change AND management; agile AND requirements AND specification.

We searched in the IEEE Xplore database (including periodicals and conference proceedings) and additionally entered the search keys in a Google search which delivered publications for instance from the ACM portal, Requirements Engineering

Journal, in Citeseer, and in many conference proceedings. Several hundreds of paper abstracts were checked and more than hundred papers read. And we analyzed some state of the art overviews, like the one of Finnegan [3] who reviews methods for RE for the development of Commercial-Off-The-Shelf software.

We **didn't find any investigations about the frequency** of the situation that in practice delta requirements are to be specified instead of complete software systems, and also no study about what specific challenges are met in this situation. We expect that such an investigation would find out that this under-researched situation in fact is very frequent and challenging in practice. That frequently delta requirements are implemented additionally to existing systems can be expected because software is known to evolve constantly, what is true for legacy systems as well as for Commercial Off-the-Shelf Software. Kiedaisch et al. [2] for the automotive domain emphasize that systems are not built from scratch. (However, they treat requirements reuse from former product generations.)

We **didn't find any work which discusses how to specify delta requirements without specifying the existing system completely in detail, or whether or when it is necessary to specify the existing system**. This topic has neither been treated explicitly by research nor has been discussed in the context of industry case studies. It seems that always the requirements on the system have been described completely. (This observation could be attributed to a lack of awareness of this problem, at least on the side of researchers.) While we did not find ready to use solutions the literature search stimulated us for the following solution ideas:

- *Reusable requirements*: For instance, it has been proposed to use domain models as requirements specification templates (then, for each project only the delta with respect to the domain model is specified) [4]. Reusable requirements or reusable components can be described and documented in a library, which demands a specification of the complete system, but a black box specification is sufficient [5].
- *Hierarchical refinement:* Describing the requirements on different degrees of granularity/ hierarchical refinement, allows specifying the existing system on a high level of granularity, which is sufficient for describing the context of the delta requirements, especially when the existing system is well known to the stakeholders involved. This is the approach chosen for our case study.
- *Change request templates*: Such templates are explicitly made for describing delta requirements, as each change request signifies a delta to a system. However, usually, they refer to the existing requirements which they apply to or which they influence, and thus require an existing specification.
- *Prototyping:* Prototyping can present the existing system as a graphical prototype, e.g. a user interface prototype. Alternatively, the existing system itself can be used like a prototype.
- *Agile software development methods:* These usually specify the system completely on a high level of granularity in order to obtain an initial set of requirements (in the form of user stories, features, etc.) which then are prioritized and realized incrementally. The task of specifying the requirements in more detail is done at implementation time. We found this to be true for Feature Driven Development [6], Scrum and Extreme Programming [7].

- *Specifying requirements in units which are independent of each other as far as possible:* This principle is followed in the agile development methods but also in other approaches. Such requirements units are called features [8], [9], super-requirements [10], or Minimum Marketable Features [11], [12].
- *Variability modelling for product lines*: In the context of product line development, one meets the challenge to describe requirements of several similar products. Thus, methods treating shared requirements and variability points are proposed. However, these methods also demand the modelling of the complete system. They avoid to model the same requirements several times by reusing their descriptions.

We conclude from our literature research that the specification of delta requirements is an under-researched area: It has neither been investigated how often such a situation appears in practice, nor which specific challenges it causes or how it can be treated successfully.

3 Dealing with Delta Requirements: The Case Study Approach

The RE approach we used in the case study (see Fig. 1) was based on TORE, because we expect that its hierarchical refinement of requirements supports saving effort when specifying delta requirements. TORE describes requirements on four levels (examples from the case study are given in brackets):

1. *Task Level*: On this level one specifies *actors* (e.g. External Engineer) and essential *tasks* (e.g. Generate Documentation) which these actors perform, as much as possible abstracting from the given IT support.
2. *Domain Level*: Looking at the tasks in more detail reveals the *activities* (e.g. Generate documents for import) users have to perform as part of their work. These activities are influenced by organizational and environmental *constraints* (e.g. NFR 'Pre-processed documents shall be marked ready'). At this level, it is determined how the work process changes as a result of the new system. This includes, in particular, the decision on which activities will be supported by the system (called *system responsibilities*) and which *domain data* (e.g. Report about import result) are relevant for the activities.
3. *Interaction Level*: On this level, the assignment of activities between human and computer is described by *use cases* (e.g. Import prepared document into system). They define how the user can use the *system functions* (e.g. Generate system data according to imported document) to achieve the system responsibilities. These have to be aligned with the *user interface* (UI) and the *interaction data* (e.g. path of document) which are input and output on the UI. The notations used are use cases, entity-relationship diagrams, system functions and the user interface structure (abstracting from the specific layout and platform of the UI).
4. *System Level*: The internals of the *application core* (e.g. class Import) and the *UI* are documented on the system level. They determine details of the visual and internal structure of the system.

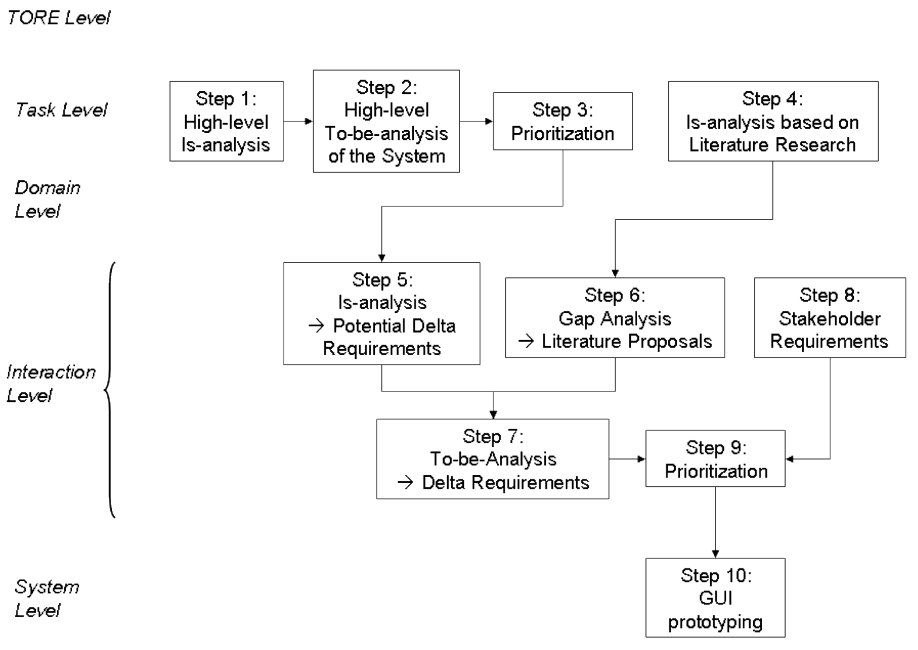


Fig. 1. Steps of our requirements elicitation and specification approach and their dependencies in terms of information flow between the steps

In the case study, the requirements were specified and prioritized by successively moving top-down from a high level of description to a lower level of description within TORE. The decision how to proceed was based on the involved researchers´ and practitioners´ experience. It was also important to integrate notations and presentations which the different stakeholders know. While the project´s technical stakeholders are familiar with use cases, the non-technical stakeholders usually discuss about software using user interface prototypes and process models like the one presented in Fig. 2. The process models contain less detail than the use cases, because they only give the steps´ names while use cases also add rules and other details about these steps.

It is important to note that in this case study the delta requirements concern change management. This should not be confused with the fact that delta requirements themselves describe changes.

The following steps were executed (see also Fig. 1):

1. *High-level as-is-analysis of what the system supports so far:* The first step was a high-level TORE analysis (task and domain level) of the existing IES in order to identify domain data and activities of the user task "change management" so far supported by IES. This resulted in 9 actors and 16 activities and roughly 50 entities covering also the system architecture.
2. *High-level to-be-analysis of the system*, i.e. identification of where in the existing system deltas are needed when integrating change management functionalities to the activities supported so far: Some of the data and activities identified

in step 1 needed to be modified in order to support change management, some others less or not at all. We identified those domain data managed by the IES which can be subject to changes and the activities which are affected by change management. This led to a matrix of four domain entities times three activities, and each of the twelve fields of these matrix was treated as a possible high-level delta requirement.

3. *Prioritization:* These twelve high-level requirements were prioritized to focus the effort of a detailed requirements specification and prototype implementation on the most important ones. In the case study the following prioritization criteria were used:
    a. The degree to which the change in an artefact affects the corresponding activity (not at all, partially, fully);
    b. For which fully affected activity tool support for change management leads to higher efficiency, because rules can be defined for the consequences which a change in the artefact has on the activity.

    In the case study, the import of customer data was rated highest and therefore chosen for detailed requirements specification and implementation. Note that customer data means IES data provided by the customer.
4. *Detailed as-is-analysis of the whole task integrated with literature research:* A high-level description of how well IES *currently supports* change management was created. As a source for desirable functionality also a literature search on tool support for change management was carried out. This description was structured according to the change management phases of ISO 10007 [13], and contains one to three sentences per phase which describe the gap, i.e. what is missing in the current system. These phases are:
    a. Initiation, identification and documentation of need for change (i.e. change request).
    b. Evaluation of change.
    c. Disposition of change.
    d. Implementation and verification of change.
5. *As-is-analysis on interaction level*: In order to analyze how the chosen activity (see step 3) is currently supported by the tool, it was refined by a use case like scenario and graphically modelled as a process (see Fig. 2) and at the same time described in textual form in a use case template. We did not use standard use case templates, because they demand much detail which was not necessary in our case study because the current processes are known to all stakeholders. Therefore, we rather talk of "scenarios" in what follows. Furthermore, the user interface was abstracted to a UI structure (specified in terms of workspaces as defined in the contextual inquiry approach [7]). This detailed as-is-analysis was the basis for the detailed delta requirement specification. The graphical process notation was already known to most of the stakeholders and used mostly for the discussion with them. The textual scenario specification was used as a means of rigour to document the common understanding of the process diagram in more detail.
6. *Gap analysis on interaction level.* The proposed requirements from literature (step 4) were scrutinized to find out whether they are relevant for the process. The literature study identified 10 such "literature proposals". Examples are

"Add representation relationships / horizontal relationship: The IES shall support relationships between X and Y", i.e. they describe what is to be changed within the system on the interaction level. They were added to the list of potential delta requirements.

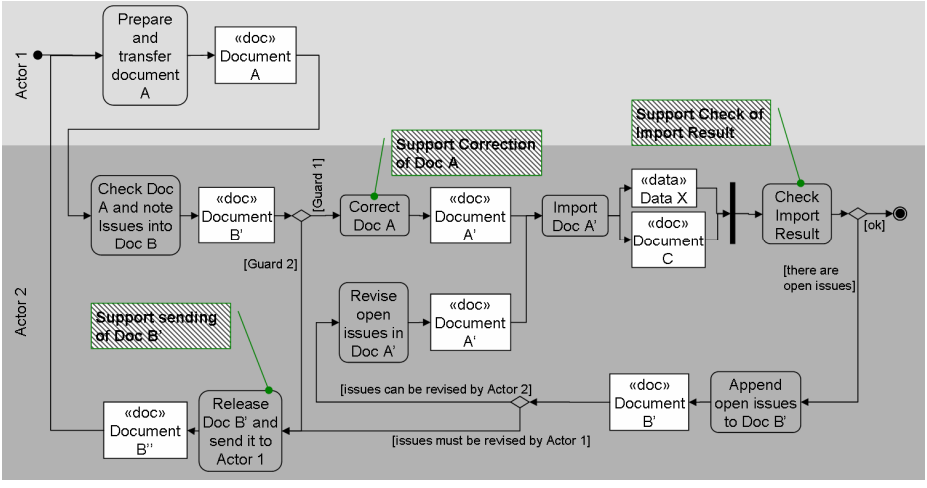


Fig. 2. An example scenario, modeled graphically as a process including potential delta requirements (hatched boxes). This model´s focus are the actor steps, not the system steps.

7. *To-be-analysis on the interaction level in order to identify proposals for delta requirements on interaction level*: For the chosen activity, ideas for implementing change management functionalities in IES were generated. For each scenario step specified in step 5, we analyzed whether it needs to be changed to support the new functionality (i.e., change management). In particular, steps which are currently executed manually were analyzed whether tool support makes sense here. Scenarios are described as high-level as possible, while those parts of the scenario which are relevant for change management, are "refined", i.e. described in more detail and highlighted by a grey (upward diagonal hatched) background. New ideas were included in the process model and the scenario text as comments and highlighted by downward diagonally hatched background. Variants of scenarios were described, not by specifying several versions of the same scenario, but by separately specifying alternatives of single steps, which were identified unambiguously by numbers. This led to 13 potential delta requirements. Some of these were further detailed with specific scenarios giving rise to 6 additional potential delta requirements. See Fig. 2 and Fig. 3 for examples of such extended scenario descriptions, both in graphical and in textual form. Fig. 4 shows an example of a variant description. For reasons of confidentiality, the examples in this paper use general wording. An example of a potential delta requirement is "support follow-up of open issue". Thus, altogether 29 potential delta requirements were collected in steps 6 and 7.

Flow of Events

This use case description bases on the Use Case X. Modified or additional steps are marked in grey (additional text is printed in italics).

Step	Actors	System
1	Actor 1 prepares Document A and hands it over to Actor 2.	
1a	*Actor 1 informs Actor 2 which changes have been introduced to Document A.*	
2	Actor 2 checks Document A	
3	Actor 2 adds entry to Document B describing details to be clarified/ explained by Actor 1 and decides whether a. Document quality is sufficient, i.e. there may be open issues but Actor 2 can continue with his work: Continue with step 5. b. Document quality is not sufficient: Continue with step 4.	(supported by Application A)
4	Actor 2 decides to release Document B and hands it over to Actor 1. Continue with step 1.	
5	Actor 2 modifies Document A to Document A'.	(supported by Application B)
6	Actor 2 starts Module M and imports Document A'.	Module M creates/updates Data X and creates Document C (include System Function 1)
7	Actor 2 reviews Data X and Document C *(he also checks the Data X for consistency, plausibility and correctness)*. He decides: a. There are no errors or the errors / warnings can be ignored: Exit use case. b. The errors / warnings must be processed: Continue with step 8. *c. There are duplicate elements in Data X* *Continue with step 8.*	(supported by Module M)
8	Actor 2 adds entry to Document B' and decides whether a. he can process the errors/ warnings *or process duplicate elements in Data X*: Continue with step 9. b. the errors/ warnings *or duplicate elements* must be processed by Actor 1: Continue with step 4.	(supported by Application A)
9	Actor 2 processes errors / warnings in Document A'. *He also revises inconsistent, implausible or incorrect data.* Continue with step 6.	(supported by Application A)

Fig. 3. Example scenarios described in a use case template, with refinements of base scenario (italic font / upward diagonal background)

8. *Elicitation of detailed stakeholder requirements*: The potential delta requirements were discussed in detail with the stakeholders. This revealed 19 stakeholder requirements on how the deltas to the current process and tool support should be carried out.

Proposals for future Use Case

The following proposals describe alternatives where these manual interactions are supported by the system. Modified or additional steps are marked in green (additional text is printed in italics).

Note: Proposals explained for the Use Case X are also applicable for this use case, but not mentioned a second time to prevent ambiguity.

P1 Support identification of duplicate Data elements

7	Actor 2 reviews Document C. *He tells the system to check Data X.*	(supported by Module M)
7a		*The System checks for duplicate Data elements and proposes elements to be deleted (include System Function 2)*
7b	Actor 2 selects: a. *The system does not propose duplicate Data elements* and Document C does contain no errors, or the errors / warnings can be ignored: Exit use case. b. *The system does propose duplicate Data Elements or* the errors / warnings must be processed. Continue with step 8.	

Fig. 4. Example for potential delta requirement (i.e. Proposals, downward diagonal background)

9. *Prioritization on interaction level*: The 29 potential delta requirements from steps 6 and 7 were mapped against the 19 stakeholder requirements from step 8. This revealed which delta requirement was how useful. Altogether, 13 delta requirements were selected for the implementation.
10. *UI prototyping for delta requirements*: For these, a detailed solution specification was derived in terms of a UI prototype and iteratively discussed with the stakeholders.

In short, our approach was a successive detailing of the to-be-analysis and as-is-analysis. By prioritization, it was decided which higher level requirements were to be described on a lower level of granularity.

4 Discussion of the Case Study Approach

Like all other RE methods, TORE is usually used for specifying complete software systems. It proceeds top-down over four levels of description, which refine the requirements successively. The exact procedure for the creation of the descriptions on the different levels is not fixed a priori and should be defined for the specific project purpose. In this case study we defined the TORE procedure consistent with our two specific objectives: namely describing delta requirements and keeping the specification effort low while taking profit of the fact that there is an existing system which does not need to be specified in detail. These two objectives led to the procedure described in the previous section. Resulting from our experiences in the above case study, these are the major extensions in the use of TORE for the purpose of delta requirement specification:

- We performed both: an as-is-analysis and a to-be analysis. The results are ideally described in the same form. This simplifies the identification of deltas between the existing process / system and the to-be-process / system. TORE so far supports the as-is- and to-be analysis on the domain level. We enhanced TORE to support this also on the interaction level.
- We executed several stakeholder sessions on the interaction level. The to-be analysis was essentially carried out in two steps: First potential delta requirements were proposed by analyzing the existing processes and system support. This could be done with the aid of only few stakeholders, mainly based on general change management knowledge. In a second step, the details of the literature analysis were used to develop potential delta requirements. Only then we involved all the relevant stakeholders to bring in their specific requirements on these potential delta requirements.
- In order to highlight the most relevant scenario steps, we introduced the following color coding:
    - In derived scenarios, such steps were highlighted by a grey background (in this paper presented as upwards hatched background) which were refined more than the rest of the scenario in order to discuss potential delta requirements (see Fig. 3, Step 1a).
    - Partially described scenarios have been used to describe selective improvements, namely potential delta requirements (see Fig. 4). These delta requirements were highlighted green (here as downwards hatched background).
- We found that system steps in the scenario text could be documented minimally: In ordinary use cases, it is important to describe the details of the system support. But when describing delta requirements, in many cases these details could be seen in the existing system. If so, we focused on describing the actor steps in detail, containing also implicit information about the corresponding system support. Thus, system steps have been described in a generic way (e.g. "Supported by Application A", see Fig. 3, Step 3).
- The fact that a system already exists allowed us to ignore some rules which are commonly observed in RE. Such a rule is that requirements, e.g. use cases, must describe the "what" but not the "how", i.e. the strict separation between requirements and design. However, when the system to be specified essentially already exists, it is more understandable for most stakeholders to describe the system in terms they know. This can mean to name technical system components, software names (trademarks) and names of the software modules used in the company. Thus, we allowed the scenarios to use system specific terms, i.e. to mention the system components which are responsible for a specific scenario step (see Fig. 3, Step 6).
- During TORE´s top-down refinement, on each level decisions are made. Such decisions can mean to prioritize the requirements, with respect to which of these are to be refined to the next level. Such an explicit prioritization – as we did on the task, domain and interaction level - is especially important when the objective is to focus the specification effort on delta requirements.
- The overall comprehension was supported by the intense use of diagrams. Such diagrams were process models for presenting the scenarios graphically (see for instance in Fig. 2) and GUI prototypes. These diagrams have been a good basis

for discussion with the non-technical stakeholders. Furthermore, it was possible to use GUI prototypes earlier in the requirements specification process than usually, and they played a more important role than usually in TORE, because the UI essentially already existed. The GUI prototype did not play the role of a specification result, but rather as a specification backbone and basis for describing deltas. They supported specifying scenarios as well as the UI itself.

5 Lessons Learned

In this section, we discuss lessons learned from the case study with respect to the specification of delta requirements, TORE´s applicability for this task and RE in general.

In this case study, we felt that it would not have been a good option to exclusively specify the delta enhancement of the system. This is because we did not simply enhance the software system incrementally, but our objective was to improve processes. Therefore, some sort of process analysis was necessary. The as-is-analysis supported the analysis of the effects of the delta requirements on other processes.

TORE facilitated the analysis of an existing system: High level requirements like actors, user tasks and workspaces of the existing system could easily be described. TORE offers guidelines for describing the RE artifacts. Furthermore it was easy to do as-is-analysis as well as to-be-analysis and to include several prioritization steps in order to identify requirements which are to be refined further.

It was easy to modify TORE, as each part of TORE has a well-defined purpose capturing a specific part of the requirements specification. We could easily identify those parts of TORE which needed to be modified and also to check what modifications would be induced on related parts of the specification.

Besides the above benefits also difficulties have been encountered. Describing the delta requirements was not as easy as the analysis of the existing system. Different stakeholders needed different notations. For the technical stakeholders, use case templates were a good means for specifying requirements, even if we did not adhere to standards, while the non-technical stakeholders preferred graphical process models. The process diagrams used in the case study were accepted and usually applied at ABB. So they were used in the discussion with the stakeholders while the scenario texts were only used by us for capturing more detail. While the UI structure was too abstract for the discussion, useful feedback was created after presenting GUI prototypes. These GUI prototypes supported discussions about and decisions on the scenario steps. This shows that GUI prototypes must be presented as early as possible. Discussions during requirements specification not only focused on the delta requirements but also on the right form of requirements specification. Reasons for this were difficulties with finding the ideal granularity of description, different usage and understanding of the terms ‘Actor’, ‘User Task’ and ‘Role’ (in the context of ABB a role is used to describe a task of one actor). This shows that some time is needed to introduce in an organization an RE specification method with new concepts.

The need was felt for means for documenting discussion and prioritization. Discussions about variants of the identified scenarios were documented separately in a text processing tool. Versioned and commented presentation slides were used to capture further information about the discussions. This includes documentation of brainstorming

activities, evolution and documentation of different versions of RE artifacts, in particular process descriptions. A possibility to tag the state of a requirement was desired (e.g. proposal in discussion, proposal accepted, and proposal postponed) and how to distinguish between obligatory and optional requirements.

Specific and flexible tool support would be helpful. In this case study, a text processing tool was used to document requirements instead of the specific TORE tool [14]. The implementation of the method modifications in the tool would have been too time-consuming (e.g. modified use case template). Using text processing resulted in additional effort (e.g. of establishing references between the artifacts), but it offered the necessary flexibility and appropriate data security.

TORE supports the specification of both functional and non-functional requirements (FR and NFR). However, in the case study we focused on FR and only to a small extent on NFR. This made sense because the realization of the case studies delta requirements did not influence the satisfaction of NFR, e.g. maintainability. We expect that this often is the case when an existing system is enhanced by small delta functionalities. However, the situation is different when the delta requirements are NFR, e.g. when one aims at improving the systems quality, e.g. to increase the systems performance.

6 Conclusion and Outlook

In an industry case study, we met the situation that an existing large software system without a specification had to be enhanced by change management functionalities. As we did not want to specify the existing system completely, we explored a way to specify the delta requirements in detail and describing the existing system less detailed. This was supported by TORE, which specifies software requirements on four levels of granularity. Two main modifications were made to TORE. The first modification is the parallel specification of the system as it is and of the system as it should be on several levels, using the same notation for both, in order to identify the gaps between them. The second modification meant to include prioritization on each specification level in order to identify the requirements which are to be specified in more detail. This approach worked well. We managed to describe delta requirements so that they could serve as the basis for their successful implementation, while saving specification effort where it was not necessary.

We were surprised to find that there is no research which explicitly treats the question how delta requirements can be specified without specifying the complete system in detail. This we find especially surprising as we expect that this challenge is frequently met in practice. We see the need for further research about the specification of delta requirements. We plan to investigate the following questions further:

- How frequent is the situation in practice that delta requirements are to be specified? How are delta requirements specified in practice?
- How much of the as-is-system must be specified in order for the delta requirements to be understandable? How well does our approach scale to other contexts?
- How well do the traditional RE methods support this?
- Do reusable requirements (e.g. quality models, domain models or requirements patterns) support the specification of delta requirements?

References

1. Dutoit, A.H., Paech, B.: Developing Guidance and Tool Support for Rationale-based Use Case Specification. In: REFSQ - Workshop on Requirements Engineering for Software Quality, Foundations for Software Quality, pp. 85–100. Springer, Heidelberg (2001)
2. Kiedaisch, F., Pohl, M., Weisbrod, J., Bauer, S., Ortmann, S.: Requirements archaeology: from unstructured information to high quality specifications [in the automotive industry]. In: Fifth IEEE International Symposium on Requirements Engineering, pp. 304–305 (2001)
3. Finnegan, R.: Requirements engineering methodologies for COTS systems. In: IEEE International Conference on Electro/Information Technology EIT, pp. 475–481 (2008)
4. Paech, B., Kohler, K.: Task-driven Requirements in object-oriented Development. In: Leite, J., Doorn, J. (eds.) Perspectives on RE. Kluwer Academic Publishers, Dordrecht (2003)
5. Boehm, B.: Requirements that handle IKIWISI, COTS, and rapid change. IEEE Computer 33(7), 99–102 (2000)
6. Palmer, S.R., Felsing, J.M.: A Practical Guide to Feature-Driven Development. The Coad Series. Prentice Hall PTR, Upper Saddle River (2002)
7. Beck, K.: Extreme programming explained. Addison-Wesley, Upper Saddle River (2000)
8. Nejmeh, B., Thomas, I.: Business-Driven Product Planning Using Feature Vectors and Increments. IEEE Software 19(6), 34–42 (2002)
9. Zhang, W., Mei, H., Zhao, H.: Feature-driven requirement dependency analysis and high-level software design. Requirements Engineering Journal 11(3), 205–220 (2006)
10. Davis, A.M.: The Art of Requirements Triage. IEEE Computer 36(3), 42-49 (2003)
11. Denne, M., Cleland-Huang, J.: Software by Numbers: Low-Risk, High-Return Development. Prentice-Hall, Upper Saddle River (2003)
12. Denne, M., Cleland-Huang, J.: The incremental funding method: data-driven software development. IEEE Software 21(3), 39–47 (2004)
13. ISO/IEC: Quality management systems – Guidelines for configuration management – ISO/IEC 10007:2003(E). International Standards Organization ISO (2003)
14. Unicase, `https://teambruegge.informatik.tu-muenchen.de/groups/unicase/`

Requirements Tracing to Support Change in Dynamically Adaptive Systems

Kristopher Welsh and Pete Sawyer

Lancaster University, Computing Dept., Infolab21 LA1 4WA Lancaster, UK
{k.welsh,p.sawyer}@lancs.ac.uk

Abstract. [Context and motivation] All systems are susceptible to the need for change, with the desire to operate in changeable environments driving the need for software adaptation. A Dynamically Adaptive System (DAS) adjusts its behaviour autonomously at runtime in order to accommodate changes in its operating environment, which are anticipated in the system's requirements specification. **[Question/Problem]** In this paper, we argue that Dynamic Adaptive Systems' requirements specifications are more susceptible to change than those of traditional static systems. We propose an extension to i* strategic rationale models to aid in changing a DAS. **[Principal Ideas/Results]** By selecting some of the types of tracing proposed for the most complex systems and supporting them for DAS modelling, it becomes possible to handle change to a DAS' requirements efficiently, whilst still allowing artefacts to be stored in a Requirements Management tool to mitigate additional complexity. **[Contribution]** The paper identifies different classes of change that a DAS' requirements may be subjected to, and illustrates with a case study how additional tracing information can support the making of each class of change.

Keywords: Adaptive Systems, Requirements Evolution, Traceability.

1 Introduction

Changes can be required of a system at any stage during its design, implementation or useful life. In traditional *static* systems, these adaptations (e.g. changed requirements), are made offline by the system developers as maintenance. Recently a new class of system has begun to emerge, capable of adapting to changes in its environment autonomously at run-time. Such *Self-Adaptive* or *Dynamically Adaptive Systems* (DASs) are designed for volatile environments where the system requirements and/or their priorities may change along with the environment even while the system is running. The nature of the problem domains for which DASs are conceived are such that their environments may be only partially understood at design time. Similarly and particularly for embedded DASs, the potential for new and exploitable technologies to emerge during the course of the system's life is high because, with the current state-of-the-art, a DAS often represents a novel application of emergent technologies. The DAS itself may exhibit emergent behaviour as it re-configures itself dynamically, in ways and under circumstances that may have been hard to anticipate at design-time.

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 59–73, 2009.

Thus, far from their runtime adaptive capability making them immune to the need for offline adaptation, DASs are particularly susceptible to it.

It has long been recognized [1] that requirements management (RM) is needed to help deal with anticipated change by recording (among other items of information) as traces the relationships between requirements and the down-stream artifacts of the development process. This allows questions about how requirements came to be and how decisions were reached to be answered later in the development process, or indeed after deployment. Given the identified need for evolvability and the variety of factors that may mandate it, the importance of traceability information in a DAS is, we claim, at least as high or even higher than in a comparable static system.

In this paper we identify requirements for traceability of DASs and, building on our earlier work on using goal models to discover DAS requirements [2], [3] show how i* [12] models can be augmented to record this information, for later integration in a Requirements Management tool.

The rest of this paper is organised as follows: Section 2 looks at related work in the area of DASs. Section 3 identifies the types of change that the specification of a DAS may need to accommodate, section 4 examines the types of traceability required to support these changes. Section 5 proposes a lightweight, i* based method of capturing these types of traceability information for the purposes of modelling the proposed DAS' adaptive behaviour. Section 6 presents a case study demonstrating the use of Section 5's method to revisit decisions in light of different types of change, whilst Section 7 concludes the paper.

2 Related Work

The requirements engineering (RE) community has recently started to investigate higher-level runtime representations that would support self-adaptation. Although the challenges posed by DASs to RE were first identified over ten years ago – principally, in run-time monitoring of requirements conformance [4] [5] [6] [7] – there are few current approaches for reasoning at runtime about system requirements.

There is a strong case that any such approach should be goal-based and a number of authors [8], [9], [10], [3] report on the use of goals for modelling requirements for DASs. This work commonly recognises that the aim of the system should be to *satisfice* its overall goals even as the environment in which it operates changes. Here, adaptation is seen as the means to maintain goal satisficement, while goal modelling notations such as KAOS [11] and i* [12] support reasoning about both functional and non-functional (*soft*-) goals. We have previously argued [2] that context-dependent variation in the acceptable trade-offs between non-functional requirements is a key indicator of problems that require dynamically adaptive solutions.

Goldsby *et. al*. [3] use i* as part of the LoREM process which partially implements the four levels of RE for self-adaptive systems proposed by Berry *et al.* [13]. The latter work is interesting because it partitions the environment into discrete *domains*, each of which represent a state of the environment with distinct requirements. The system's adapted configuration for each domain is termed a *target system*. LoREM has validated this approach on the requirements for a self-adaptive flood warning system implemented in the GridKit adaptive middleware system [14] which we use in

this paper as a case study to illustrate our approach to tracing. To date, DASs have not attracted any special attention that we are aware of from the RE research community for the challenges they pose to RM and tracing.

Ramesh and Jarke [15] categorise users of traceability information into two groups: *high* and *low*-level, depending on the types of use made of the information. Two traceability uses are of particular interest to DASs: traceability of decision rationale and for evolvability. To allow decision rationale tracing, a record needs to be made of all the viable alternatives considered, along with assumptions of the impact the selection of each alternative would have. Typically, this sort of information is only kept by so-called high-level traceability users, working on large, complex projects; with patchy coverage of rejected alternatives. Requirements evolution can occur due to a change in user needs, or due to an identified deficiency in the system. As requirements at different levels of detail change, the question "Where did this requirement come from?" becomes harder to answer. By explicitly recording changes to requirements, it becomes possible to understand how the system has evolved (or needs to) over time, and to identify derived requirements that need to be revisited in the light of new changes. Given the increased likelihood of change impacting a DAS' requirements, the likelihood of encountering derived requirements whose necessity is hard to establish, or even understand is also, we argue, increased.

Furthermore, given that these requirements may form the basis of the justification for several decisions, each of which will be repeated (with different environmental assumptions) for several target systems, a change in requirements can have a widespread impact. The likelihood of far-reaching change impact means that traceability of decision rationale and for evolvability are crucially important for DAS developers as a consequence. Therefore, we argue that DASs promote high-level traceability as essential practice.

3 Types of Change

Although a DAS can adjust its behaviour in response to change whilst in operation, this does not mean that all changes can be handled automatically by the system. New requirements, new technology or simply a better understanding of the environment may all require the system to be re-specified, in whole or in part. If already deployed, the system will need to be taken offline for static adaptation to be carried out. The static adaptation process is not radically different to that of a traditional (non-adaptive) system, but the relative complexity of a DAS coupled with the increased likelihood of change means that an inefficient change management process will rapidly become problematic.

Our work [2] [16] builds upon Berry *et. al.*'s four levels of RE for Dynamic Adaptive Systems [13]. Berry *et al* start with the assumption that the environment in which a DAS must operate can be characterized as discrete states or *domains*. Each domain is served by a separate *target system*, which in a DAS is a conceptualization of a system configuration. Hence, when the environment makes the transition for domain 1 to domain 2, the DAS adapts from target system S1 to target system S2.

We use i* [12] to model the system at each level. Level 1 RE is done on a per-target system basis, specifying how the system functions in each. Level 2 RE specifes

which target system to adopt given the runtime context. Level 3 RE specifies the requirements on the system's adaptation mechanism, and how it achieves its level 2 RE. Level 4 RE is the most abstract level, covering adaptation mechanisms in general. Most changes will involve some modification to the level 1 models, which is the focus of this work. In cases where a previously made decision is changed, the system's level 2 and 3 models may also be affected; these secondary impacts are the subject of ongoing work.

We have identified five distinct classes of change that a DAS' specification may need to be adjusted for. Each needs handling differently.

- **Environmental Change.** This will be particularly common given the inherent volatility of a DAS' proposed environment; which increases the likelihood of individual domains being misunderstood, or entirely new domains being discovered. This class of change may occur during any stage of the system's life cycle, and the need for change may be punctuated with a system failure if it emerges only after deployment. A change to a previously identified environmental domain will trigger the need to re-evaluate the design decisions taken for it, whereas the identification of a new domain will require a new target system (and associated model) to be created, possibly based on that of another target.
- **Broken Assumption.** This may be an assumption about the environment in a given domain ("There will be ample solar power during the day"), or an assumption about a given system component's suitability for a domain ("Communicating via Bluetooth will use less power than Wi-fi"). The assumptions underpinning the level 2 modelling ("The environment will not switch directly from S1 to S6" or "The system will spend most of its time in S1") are also classed as environmental assumptions, and will affect several different levels of model. Assumptions such as these may be broken as designers better understand the domain or the proposed system, or may only become apparent after deployment. An assumption being broken triggers the need to re-evaluate all decisions based upon it.
- **New Technology.** The availability of a new technology that can be exploited can be modelled as a new alternative for a decision. Given the relative immaturity of adaptation frameworks this will likely occur frequently. As with a static system, designers need to weigh the potential costs and benefits to decide whether to take advantage of the new technology or not. However, for a DAS the designers will need to make the decision for each target system. If the new technology is utilised in one or more targets, other decisions that impact on the same quality features as the new technology in these targets will need to be revisited.
- **Consequential Change.** This is so named because the change is necessitated as a consequence of a previous change. This kind of change will be particularly important in systems with a budgeted requirement such as a maximum power consumption or maximum total weight. In this case, making any of the previous types of change can require a re-evaluation of previous decisions across all domains, trying either to bring the system back down to budget if the change negatively impacted the budgeted requirement, or to more fully utilise the budget if the change created headroom.

- **User Requirements Change.** This class of change is of course not specific to DASs. However, the impact may reach several target systems, essentially multiplying the workload involved in making the change. User requirement changes are difficult to predict, and are also variable in the extent of their impact.

4 Traceability Requirements

Given the increased likelihood of change impacting the requirements specification of a DAS, and the fact that decisions are essentially taken again and again for each target system, traceability information becomes more important. Each of the types of change discussed in the previous section has differing traceability requirements.

- **Environmental Change.** When environmental understanding changes, an entire model will need to be reconsidered, or created from scratch if a new domain has been discovered. Essentially, this will involve re-making all the per-domain decisions in light of the changed (or new) environmental knowledge. For this type of change, the only traceability information needed is the ability to identify all the decisions taken for a given target system. In most cases, the target systems of a DAS will share much commonality, perhaps exhibiting variability only in a small number of discrete components. As such, each's level 1 model will merely be a refinement of another's.
- **Broken Assumption.** When an assumption has been broken, either by improved understanding of the environment or available alternatives, or having been broken demonstrably in the field, all of the decisions based on this assumption will need to be revisited. In order to facilitate this forward tracing, there needs to be a record of all decisions reliant on this assumption, along with the information on alternatives required to re-make them.
- **New Technology.** A new decision alternative, often brought about as a result of newly available technology, will require only a limited number of decisions to be revisited. However, each decision may need to be revisited for each target system. As such, to support this backwards tracing, there needs to be a record of alternatives previously considered and what basis the previous decision was taken on.
- **Consequential Change.** A consequential change, so named because the change is necessitated as a consequence of a previous change, requires the ability to trace across target systems all decisions made that affected a given, budgeted requirement. As such, there needs to be a record of which requirements are affected by which selection alternatives, to allow the analyst to search through the system to find an acceptable change to trade-off against the previous, necessitating change.
- **User Requirements Change.** A user requirement change can potentially affect any or all target systems, or may necessitate a change to a static (i.e. identical in all targets) component of the system. Therefore, as with user requirement changes to static systems, the impact of this type of change varies from case to case.

From the range of traceability requirements above we can conclude that for decisions to be re-visited and re-evaluated it is necessary to record all of the alternatives considered, along with the rationale behind the selection made. Each alternative requires assumptions that impact on the system's competing requirements to be recorded, along with the presumed impact itself. The relative importance of the system's competing requirements in this domain drives per-target selection decisions, and should be recorded along with the environmental assumptions that underpin the re-prioritisation.

5 Recording Traceability Information

In our earlier work [3] we argue that DAS development should commence with an early goal modelling phase that allows the analyst to reason about the environment, the goals that the actors within that environment have and the softgoals that constrain the quality with which the goals can be satisfied. We use the i* modelling language [12] in our LoREM [16] method for developing the requirements for DASs since i* is well matched to these aims.

There are two types of i* model: the Strategic Dependency (SD) and Strategic Rationale (SR) models. The SD model is intended to identify the goals and softgoals in complex, multi-agent systems and the dependencies between agents engaged in attempting to satisfy the system's goals and softgoals. The SR model is intended to explore the available alternatives for satisfying each agent's goals and softgoals. The SR models offer a useful means of reasoning about trade-offs, primarily in terms of the extent to which the various solution alternatives satisfy the softgoals. As the "rationale" in SR suggests, SR models serve to record the rationale for decisions. Hence, i* can be thought of as not only a modelling technique, but also as a means for tracing the motivation for requirements. It is this feature that we propose to exploit here.

However, although an i* SR model allows us to infer an actor's basis for making a decision by examining the decision's impact on system goals, the conveyed information on impacts is limited, and understanding complex decisions that balance conflicting goals, such as most adaptation decisions, is difficult. The NFR Framework [17] to which i* is related includes a mechanism for recording beliefs and assumptions, referred to as *claims*. A claim can be used to support a decision in two ways: attached to a node on an NFR model, it represents a comment about an alternative, which may influence its selection; attached to a link, it represents a comment on the importance (not the magnitude) of the impact. By adding claims to i* SR diagrams, it is possible to convey similar information, allowing decision rationale to be inferred more effectively.

A claim may be used to explain a decision alternative's negative impact on the system softgoal "minimise running costs" by claiming that this alternative "requires an additional operator for monitoring". On a standard SR diagram, by contrast, only the negative impact itself would be recorded, with the rationale merely implicit and thus potentially inexplicable to a developer tasked with evolving the system. A claim could also be used to "de-prioritise" a high magnitude impact. For example encrypting an intranet site may significantly hinder scalability, but still be deemed necessary for security, despite the positive impact on the security goal being weaker.

In most instances, it is possible to record a decision justification using a claim in two ways. The first is to invalidate a selected candidate's negative impact(s) by claiming they are insignificant or unavoidable, or to promote an alternative's claiming them too costly. The second is to promote a candidate's positive impact(s) by claiming them necessary, or to invalidate an alternative's claiming them needless. Although both are equivalent when revisiting decisions later, and those with scalability concerns may wish to record the rationale using the fewest additional entities, we prefer to record whichever way is closest to the rationale actually used. Figure 1 shows an example of an NFR claim being used to justify a decision using the first method: promoting the rejected alternative's negative impacts. The selected alternative is coloured white for clarity.

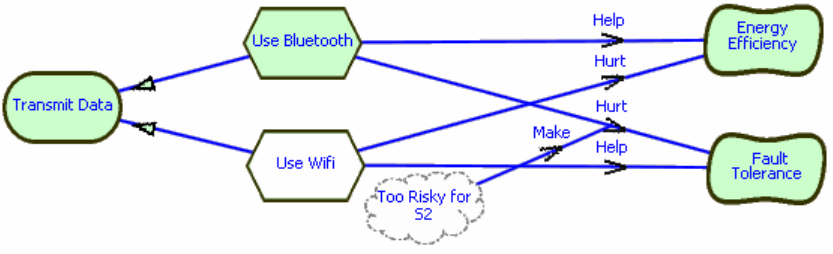


Fig. 1. NFR claim justifying selection of a component negatively impacting Energy Efficiency

In this example, the DAS under analysis is a wireless sensor network and one of its goals is to "Transmit data". This has spurred the need to select a wireless communication standard and two alternatives have been identified, modelled as: *Use Bluetooth* and *Use WiFi* (IEEE 802.11). When alternatives such as these exist, reasoning about which one to select is aided by considering their impact on softgoals. In Figure 1, two softgoals have to be satisfied: "Energy efficiency" and "Fault tolerance". These two softgoals are imperfectly compatible, thus spurring the kind of high-level trade-off analysis that i* is so useful for.

The relatively short-range Bluetooth solution is energy efficient but its ability to tolerate a node failure is constrained: since if data is routed via nodes A and B, and node B fails, the next closest node may be too remote to establish a connection with A. The WiFi alternative, by contrast, endows the nodes with greater communication range at the cost of (relatively) high power consumption. These relative strengths and weaknesses are captured crudely by the "helps" and "hurts" *contribution links* between the alternative solutions and the softgoals. It is these contribution links that we propose should be annotated with claims.

As the environment cycles through its set of possible domains, the DAS needs to adapt to the appropriate target system. Analysis of the environment shows that much of the time, the DAS will operate in a benign domain in which the risk of node failure is low. However, another domain exists, for which the target system is labeled "S2", where the risk of node-failure is significant. A claim attached to the contribution link between the "Use Bluetooth" task and the "Fault tolerance" softgoal records the rationale for why the extent to which selecting Bluetooth as the communication standard hurts fault tolerance makes its selection unacceptable for S2.

Note that this is different from using fine-grained contribution links (*Make*, *Break*, *Some*+, ++, etc.) because although defining impact magnitude, the weight (importance) of the contribution link is context- (i.e. domain/target system) dependent. Nor are claims the same as i* "beliefs" which represent assumptions made by an actor, with no presumption of truthfulness by the analyst. In the next section we expand upon the wireless sensor network example.

6 Case Study

GridStix [19] is a DAS built to perform flood monitoring and prediction, and is deployed on the River Ribble in North West England. It takes the form of an intelligent wireless sensor network, with multiple nodes measuring river depth and flow rate using a variety of sensors, including the analysis of images taken with an on-board digital camera. The nodes have processing capability and memory. This allows processing of the data and execution of the predictive models of the river's behaviour to be performed on site with the system acting as a lightweight grid. However, the nodes are resource-constrained and some tasks are best performed by distributing computation among the nodes. Distributing computation has a cost in terms of the power consumed by inter-node communication, however. This is a serious issue since GridStix's location is remote and power has to be provided by batteries and solar panels, which provide only limited power.

Domain experts identified three distinct environmental domains that GridStix needs to operate in. In the "quiescent" domain, the river has a low depth and flows relatively slowly. In the "high flow" domain, the river flows faster, but is still at a relatively low depth, this can presage the rapid onset of the third domain "flood", where the river depth has started to increase, and there is imminent danger of flooding, which poses a danger to both local residents and the system.

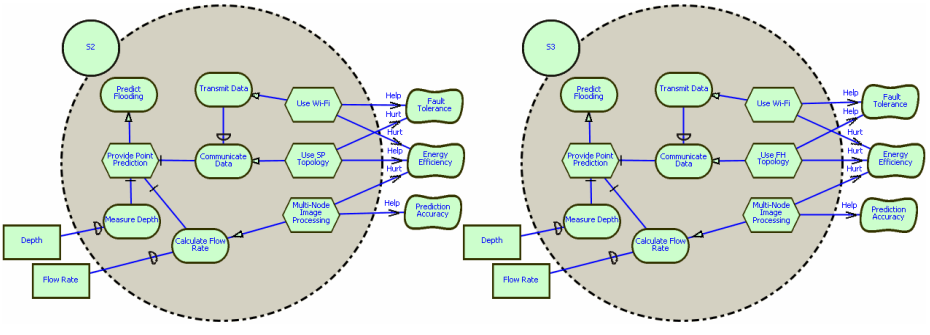


Fig. 2. Models of GridStix configured for *High Flow* (S2) and *Flood* (S3) domains

GridStix's key softgoals are "Energy efficiency" to maximise battery life, "Prediction Accuracy" to provide timely and accurate flood warnings, and "Fault Tolerance" for survivability. The system is built upon the Gridkit middleware [14] which allows software components to be substituted at runtime. Our practice is to develop a separate SR diagram for each target system. Figure 2 shows part of those for target system

S2, which operates in the high flow domain, and S3 which operates in the flood domain. The actual, deployed system varies a wider set of individual components dynamically, but their inclusion would increase the models' complexity for little or no illustrative benefit.

The S2 and S3 SR diagrams illustrate which decisions were reached for each target system. Notice that the only difference between S2 and S3 is that in S3, the task "Use FH topology" substitutes for "Use SP topology" in S2. SP and FH represent spanning tree algorithms used to communicate data among the nodes in the network: shortest path and fewest hop, respectively. The characteristics of the spanning tree algorithms are different and this is reflected by their respective contribution links to the Fault tolerance and Energy efficiency softgoals.

From the contribution links from the two spanning tree tasks in S2 and S3, it is possible to infer that the "Energy efficiency" softgoal was de-prioritised between S2 and S3. It is not possible, however, to revisit the decisions in light of new information or some other change, given that the alternatives considered are not documented, and that the only justification for the decision recorded is that: "This alternative helps the Prediction Accuracy softgoal, which must have been prized more highly than Energy Efficiency." The real rationale for the choice of spanning tree algorithm has been lost, *viz* in the h*igh flow* domain the risk of immersion or water-borne debris destroying a node is slight, so the need to conserve energy in case the situation worsens takes priority. The relatively energy-efficient shortest path algorithm is therefore the better choice. In the *flood* domain, however, node damage is a real risk and the relatively power-hungry but resilient fewest hop algorithm is used to favour *Fault tolerance* over *Energy efficiency*.

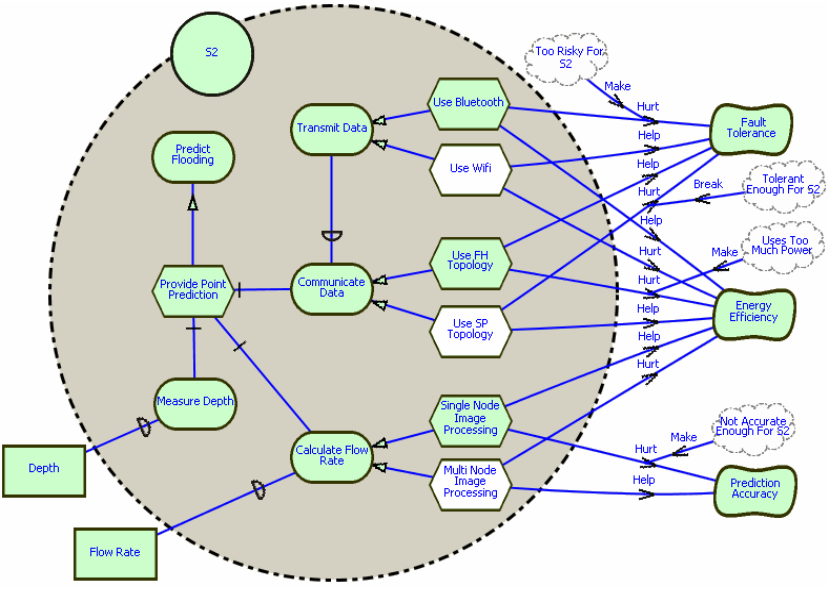


Fig. 3. Augmented model of GridStix in the *High Flow* (S2) domain

Although the limited traceability available from Figure 2 is far better than nothing, dealing with numerous, frequent specification changes and having to adjust several target's specifications using only this information is a bleak prospect. Recording the rejected alternatives and a small amount of additional information using NFR claims could make these changes far more manageable. Figures 3 and 4 show the same two models, with this extra information recorded.

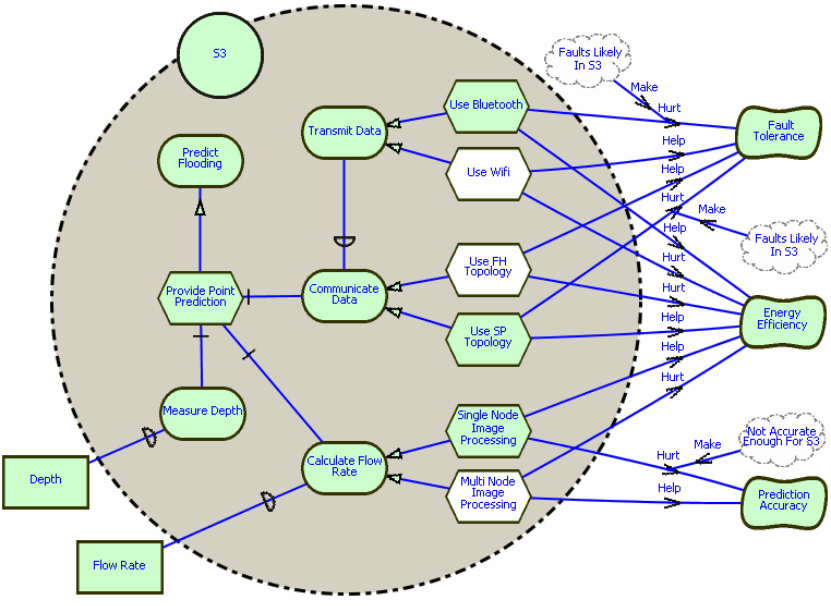


Fig. 4. Augmented model of GridStix in the *Flood* (S3) domain

The additional information shown in Figures 3 & 4 allows rejected alternatives to be re-examined if necessary with a richer (although still incomplete) understanding of the previous decision's basis. We believe that the information recorded in Figures 3 & 4 is the minimum required to allow per-target system decisions to be revisited.

In section 3, we identified five distinct classes of change that a DAS' specification may need to be adjusted for. We now discuss how recording additional traceability information allows some of these types of changes to be carried out more easily.

Environmental Change. The first type of change identified was Environmental change. This could take the form of a new environmental constraint, an adjustment to domain boundaries, or an entire domain being introduced, eliminated, or merged with another. We illustrate the new domain scenario, by introducing a fourth domain to the GridStix system. This fourth domain, blockage occurs when a natural or man-made obstruction hinders the flow of the river downstream from the GridStix nodes.

Although this fourth domain does not necessarily pose a danger to local residents and should not trigger a flood alert, there is a significant risk of nodes becoming submerged and quickly failing. This makes the more power efficient shortest path (SP) networking algorithm and Bluetooth communication risky choices. Furthermore, if the blockage is removed (or breached) suddenly, the river's behaviour could be very

difficult to predict without data downstream of the blockage. Therefore, it would be desirable for the system to report this condition to the Environment Agency, who would typically seek to remove occluding objects maintaining the river's regular flow. The blockage domain is characterised by a high river depth and low flow rate. It could be argued that the new domain and the requirement to report it are two separate changes (one environmental change and one user requirement change), but we have modelled both together for brevity. Figure 5 shows the SR model for the target system, S4, for the blockage domain.

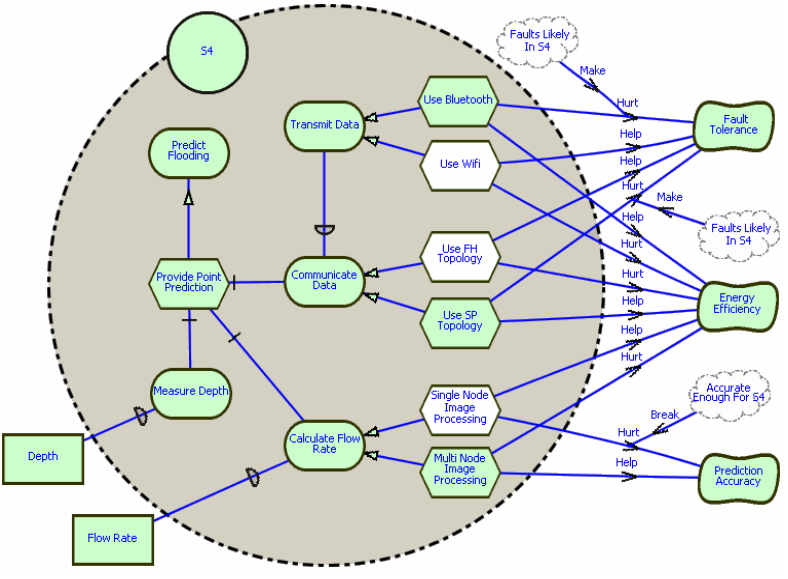


Fig. 5. Augmented model of GridStix configured for the *blockage* (S4) domain

The decisions taken for the Blockage (S4) domain closely mirror those taken for the Flood (S3) domain, with both Bluetooth and Shortest Path networking rejected on the grounds that they each compromise fault tolerance unacceptably, and that there is a real risk of node failure in this domain. Unlike the Flood (S3) domain, the chances of a sudden increase in flow rate or depth are remote, and the slower, more power efficient single node image processing algorithm can be used to save power.

Broken Assumption. The second class of change identified was *broken assumption*. This can happen as understanding of the operating environment improves, or as knowledge of available components becomes more complete. Dealing with this type of change involves tracing all the decisions made based on the assumption, and revisiting them.

To illustrate this type of change, we have modified our model of the Flood (S3) domain, replacing the assumption "Single node image processing is not accurate enough for S3" with the inverse. In this instance, changes are confined to this model, although this will not always be the case. In fact, only one decision is reached on the basis of this assumption, and Figure 6 shows single node image processing being used instead, along with the replaced assumption.

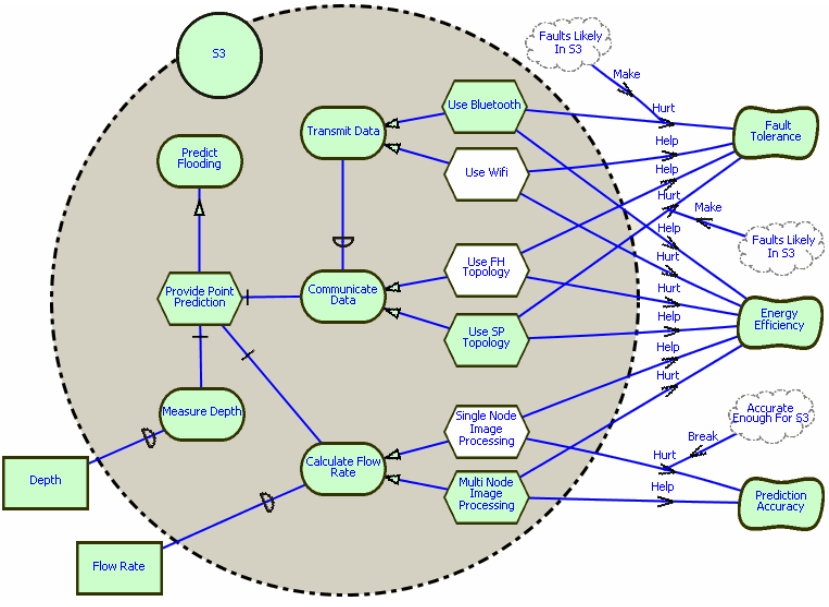


Fig. 6. Augmented model of GridStix in the *Flood* (S3) domain after a broken assumption

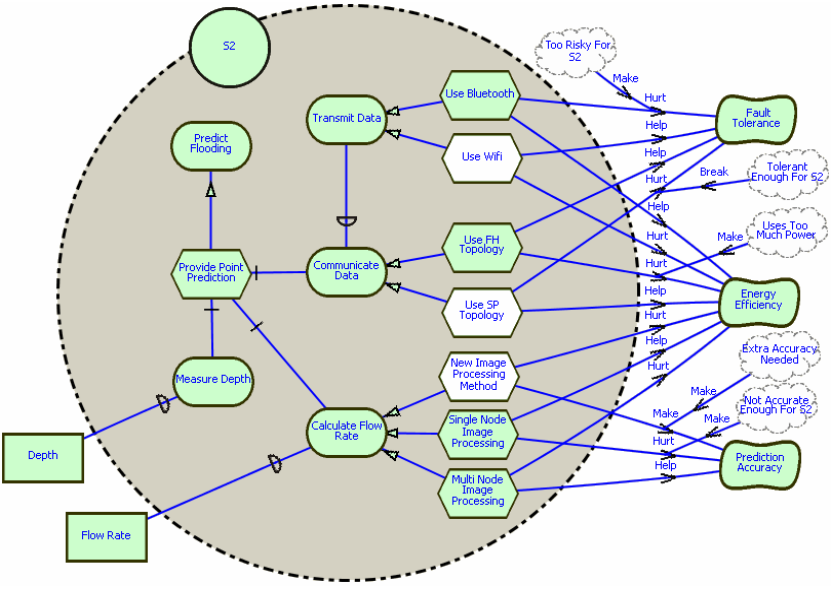


Fig 7. Augmented model of GridStix in *High Flow* (S2) domain, with new decision alternative

This class of change is particularly amenable to analysis in a requirements management tool, providing the tool has a record of decisions reliant on the now false assumption. By automating the location of affected decisions, the overhead of dealing with this class of change will be decreased, particularly in larger systems.

New Technology. The third class of change identified was a *new technology* that becomes available for use. This may introduce a brand new decision to be taken for each domain, or add a new alternative to a decision already taken. It is this second variant that we have chosen to illustrate in Figure 7, which shows the prediction model decision after being re-taken with a newly available alternative.

Although Figure 7 shows just the *High Flow* domain, the new alternative would need to be added to each diagram, creating a significant overhead in systems with many domains. The new/changed decision in each needs to be re-examined and remade in light of the new alternative, which may be selected in some, all or none of the target systems. In this example and domain, the extra accuracy offered by the new image processing method was deemed necessary and its negative impact on energy efficiency tolerated on the basis of this claimed need.

7 Conclusion

In this paper we have argued that, far from removing the need for off-line adaptation, DASs are inherently susceptible to the need to adapt in ways that are beyond the scope of their (necessarily limited) self-adaptive capabilities, which are limited in scope at design time. In practical terms, to give a DAS usefully long service life, it is likely to need to be maintained by human developers to correct defects and to take advantage of new knowledge and new capabilities. As such, a DAS' requirements specification needs to be as amenable to change as the system itself.

We have classified some of the types of change that a DAS is subject to and argued that some are special to DASs. From this analysis we have identified a need to trace how these change types impact on the requirements. To evaluate this, we have applied three of the types of identified change (Environmental Change, Broken Assumption and New Technology) to a case study we have used in our earlier work [2], [3], [16]. In this earlier work we developed a process for analysing DASs based on goal modelling using i*. We have therefore proposed recording rejected decision alternatives alongside the accepted option in i* Strategic Rationale models, which would previously have shown only the selected alternative. We have also proposed extending i* with the notion of *claims* which we have borrowed from the related NFR framework.

Claims permit us to annotate the contribution links used in i* Strategic Rationale models with the rationale underpinning a decision, explaining how it was reached in terms of the softgoals affected. The annotation also makes explicit the re-prioritisation of softgoals between domains, which previously had to be inferred by comparing several target system's models and examining differences in contribution links.

Ultimately, we envisage a target system's decision alternatives, their presumed impact on system softgoals, the selected option and the rationale underpinning the selection decision itself (recorded as claims) being mapped into a conventional tracing tool suck as DOORS [18]. Such a tool would allow the now-possible tracing to be automated, bringing greater benefit in terms of efficiency with scale.

References

1. Gotel, O., Finkelstein, A.: An analysis of the requirements traceability problem. In: Proceedings of the International Conference on Requirements Engineering, Colorado Springs (1994)
2. Welsh, K., Sawyer, P.: When to Adapt? Identification of Problem Domains for Adaptive Systems. In: Paech, B., Rolland, C. (eds.) REFSQ 2008. LNCS, vol. 5025, pp. 198–203. Springer, Heidelberg (2008)
3. Goldsby, J., Sawyer, P., Bencomo, N., Cheng, B., Hughes, D.: Goal-Based Modelling of Dynamically Adaptive System Requirements. In: Proceedings of 15th IEEE International Conference on Engineering of Computer-Based Systems, Belfast, Northern Ireland (2008)
4. Fickas, S., Feather, S.: Requirements Monitoring in Dynamic Environments. In: Proceedings of the Second IEEE International Symposium on Requirements Engineering, York, England (1995)
5. Savor, T., Seviora, R.: An approach to automatic detection of software failures in realtime systems. In: IEEE Real- Time Tech. and Appl. Sym., pp. 136–147 (1997)
6. Feather, M., Fickas, S., van Lamsweerde, A., Ponsard, C.: Reconciling system requirements and runtime behavior. In: Proceedings of the 9th International Workshop on Software Specification and Design (1998)
7. Robinson, W.: A requirements monitoring framework for enterprise systems. Requirements Engineering 11(1), 17–41 (2006)
8. Yu, Y., Leite, J., Mylopoulos, J.: From goals to aspects: Discovering aspects from requirements goal models. In: Proceedings of the 12th IEEE International Conference on Requirements Engineering (2004)
9. Lapouchnian, A., Liaskos, S., Mylopoulos, J., Yu, Y.: Towards requirements-driven autonomic systems design. In: Proceedings of 2005 Workshop on Design and Evolution of Autonomic Application Software, St. Louis, Missouri, USA (2005)
10. Yu, Y., Mylopoulos, J., Lapouchnian, A., Liaskos, S., Leite, J.: From stakeholder goals to high-variability software design. Technical report csrg-509, University of Toronto (2005)
11. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Sci. Comput. Program. 20, 3–50 (1993)
12. Yu, E.: Towards modelling and reasoning support for early-phase requirements engineering. Requirements Engineering. In: Proceedings of the Third IEEE International Symposium on Requirements Engineering (1997)
13. Berry, D., Cheng, B., Zhang, J.: The four levels of requirements engineering for and in dynamic adaptive systems. In: Proceedings of the 11th International Workshop on Requirements Engineering: Foundation for Software Quality (2005)
14. Coulson, G., Grace, P., Blair, G., Cai, W., Cooper, C., Duce, D., Mathy, L., Yeung, W., Porter, B., Sagar, M., Li, W.: A component-based middleware framework for configurable and reconfigurable Grid computing: Research Articles. Concurr. Comput. Pract. Exper. 18(8), 865–874 (2006)

15. Ramesh, B., Jarke, M.: Toward reference models for requirements traceability. IEEE Transactions on Software Engineering 27(1), 58–93 (2001)
16. Sawyer, P., Bencomo, N., Hughes, D., Grace, P., Goldsby, H.J., Cheng, B.H.: Visualizing the Analysis of Dynamically Adaptive Systems Using i* and DSLs. In: Proceedings of the Second international Workshop on Requirements Engineering Visualization (2007)
17. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-functional requirements in software engineering. Kluwer Academic Publishers, Dordrecht (2000)
18. Quality Systems & Software Ltd., Oxford Science Park, Oxford, U.K., DOORS Reference Manual (V3. 0) (1996)
19. Hughes, D., Greenwood, P., Coulson, G., Blair, G.: GridStix: supporting flood prediction using embedded hardware and next generation grid middleware. World of Wireless, Mobile and Multimedia Networks (2006)

Early Identification of Problem Interactions: A Tool-Supported Approach

Thein Than Tun, Yijun Yu, Robin Laney, and Bashar Nuseibeh

Department of Computing
The Open University
Walton Hall, Milton Keynes
{t.t.tun,y.yu,r.c.laney,b.nuseibeh}@open.ac.uk

Abstract. [**Context and motivation**] The principle of "divide and conquer" suggests that complex software problems should be decomposed into simpler problems, and those problems should be solved before considering how they can be composed. The eventual composition may fail if solutions to simpler problems interact in unexpected ways. [**Question/problem**] Given descriptions of individual problems, early identification of situations where composition might fail remains an outstanding issue. [**Principal ideas/results**] In this paper, we present a tool-supported approach for early identification of all possible interactions between problems, where the composition cannot be achieved fully. Our tool, called the OpenPF, (i) provides a simple diagramming editor for drawing problem diagrams and describing them using the Event Calculus, (ii) structures the Event Calculus formulae of individual problem diagrams for the abduction procedure, and (iii) communicates with an off-the-shelf abductive reasoner in the background and relates the results of the abduction procedure to the problem diagrams. The theory and the tool framework proposed are illustrated with an interaction problem from a smart home application. [**Contribution**] This tool highlights, at an early stage, the parts in problem diagrams that will interact when composed together.

Keywords: Problem Composition, Problem Interactions, Problem Frames, Event Calculus.

1 Introduction

One general approach to problem solving in requirements engineering is to decompose complex software problems into simpler familiar problems [1]. A software problem refers to the challenge of specifying a software system that satisfies an expressed user requirement [2]. In this approach to problem solving, no provision is initially made about the questions of if and how the subproblems obtained can be composed to solve the larger complex problem. Only when subproblems have been solved, are the concerns for composition considered and addressed as separate problems in their own right. This deferral of the concerns for composition is seen as an effective way of managing complexity in software development.

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 74–88, 2009.

However, when solutions to subproblems are found, several questions arise [3]: Are the problems free from interactions? If they interact, how do they interact? If there are undesired interactions, what can be done to remove them? In this paper, we are primarily concerned with the second question.

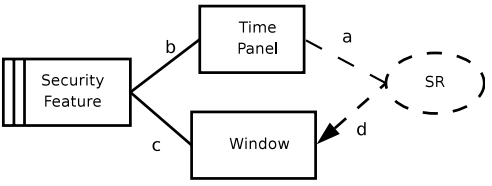
We will consider this question within the framework of the Problem Frames (PF) approach [2], which has been recognised as providing a core ontology for requirements engineering [4]. Following this approach, complex problems are decomposed by fitting their subproblems into known problem patterns. Subproblems, when composed together, may interact in unexpected ways. By problem interactions, we refer to situations where the composition of solutions to subproblems does not constitute a solution to the complex problem (from which subproblems were obtained). These interactions may be related to the issues of *consistency*, *precedence*, *interference* and *synchronisation* [2].

Checking whether subproblems in an event-based reactive system can be composed can only give event sequences where the composed requirement is satisfied (if it can be satisfied at all): it cannot tell us event sequences where the composed requirement is not satisfied. One way to solve this problem is to monitor the runtime behaviour of the system, and diagnosis failures whenever they are detected [5]. However, event sequences are identified only after the occurrence and detection of failures.

The main contribution of the paper is a tool-supported approach that uses abductive reasoning [6] to identify all possible event sequences that may cause the composition to fail. Given a description of system behaviour, an abductive procedure [7] obtains all event sequences that can satisfy a requirement, if the requirement is satisfiable. Otherwise, the procedure will report no event sequence.

In order to identify event sequences leading to a failure in composition, we negate the conjunction of all requirements to be composed as the requirement to satisfy, and attempt to abduce all possible event sequences for the negated requirement. In other words, we assume that the problems cannot be composed and ask an abductive reasoner to find out why. If the procedure returns no event sequence, there is no interactions between problems. On the other hand, any event sequence returned by the abduction is a possible interaction in the composition, and may require further analysis. Our use of logical abduction is reminiscent of [8]. In this paper, we will focus on the identification of possible interactions, whilst the issue of how undesired interactions can be removed is discussed in [9].

We have implemented a tool chain called the OpenPF to demonstrate how event sequences leading to failures in composition can be detected using the technique suggested. The front-end of the OpenPF helps requirements engineers create problem diagrams and describe the elements in the diagram using the Event Calculus, a form of temporal logic [10,11]. The back-end of our tool encodes the input into an executable specification for an off-the-shelf Event Calculus abductive reasoner [12,13] to check whether the new diagram can be composed with the existing ones. If event sequences for possible failures are found, the tool takes the abduction results and relates them back to relevant problem diagrams.



Interface Phenomena Set
TiP!a {Night, Day}
TiP!b {NightStarted, DayStarted}
SF!c {TiltIn, TiltOut}
Win!d {Open, Shut}

Fig. 1. Problem Diagram: Security Feature

The rest of the paper is organised as follows. In Section 2, we present an brief overview of the Problem Frames approach and the Event Calculus, and explain how they are used in this paper. Our approach to identifying interacting problems is explained in Section 3, and application of the OpenPF tool is discussed in Section 4. Related work can be found in Section 5, whilst Section 6 provides some concluding remarks.

2 Preliminaries

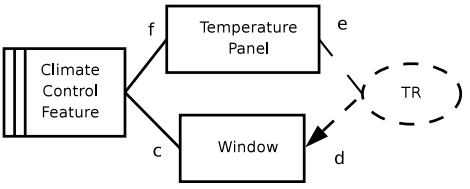
In this section, we introduce two decomposed problems related to the security feature and climate control feature of a smart home application [14]. The purpose of the simple example is to help us illustrate the approach and tool-support. An overview of Problem Frames, the Event Calculus and how we use minimal Event Calculus predicates to describe problem diagrams are also explained.

2.1 Problem Diagrams and Their Descriptions

An important feature of the Problem Frames approach (PF) is that it makes a clear distinction between three descriptions: the *requirements* (R), the *problem world domains* (W) and the *specification* (S). Broadly speaking, the requirements describe the desired property of the system, the problem world domains describe the given structure and behaviour of the problem context, and the specifications describe the behaviour of the software at the machine-world interface [2].

Security Feature in Smart Home. The problem diagram of the security feature (SF), shown in Figure 1, emphasises the high-level relationship between the requirement, written inside a dotted oval, the problem world domains, denoted by plain rectangles, representing entities in the problem world that the machine must interact with, and a machine specification, denoted by a box with a double stripe, implementing a solution to satisfy the requirement.

The problem world domains in the context of the security feature are Window (Win) and Time Panel (TiP). When describing their behaviour, we will use labels such as (Win1) and (TiP1), and refer to them later in the discussion. The window has two *fluents*, or time-varying properties, Open and Shut, each a negation of the other. At the interface SF!c, the machine SF may generate instances of events



Interface Phenomena Set
TeP!e {Hot, Cold}
TeP!f {HotStarted, ColdStarted}
CCF!c {TiltIn, TiltOut}
Win!d {Open, Shut}

Fig. 2. Problem Diagram: Climate Control Feature

(or simply *events* henceforth) `TiltIn` and `TiltOut`, which are only observed by the window (Win). The behaviour of the window is such that once it observes the event `TiltIn`, it will soon become shut (Win1); once it observes the event `TiltOut`, the window will become open (Win2), when it starts to tilt in, it is no longer open (Win3), and when it starts to tilt out, it is no longer shut (Win4), and the window cannot be both open and shut at the same time (Win5).

Similarly, the Time Panel domain has two opposing fluents, Day and Night. When time switches from day to night, the panel generates the event `NightStarted` once at the interface TiP!b, observed by the machine SF (TiP1). Likewise, when the daytime begins, `DayStarted` is generated once (TiP2). Solid lines between these domains represent *shared phenomena*: for example, c is a set of the phenomena `TiltIn` and `TiltOut`, and SF!c indicates that these phenomena are controlled by the security feature and are observed by Window. In such event-based systems, states of the problem world domains are represented by *fluents*, whilst these domains communicate by sending/receiving *events*. This is a neat mapping to the Event Calculus ontology, as we shall see later.

The requirement for the security problem (SR) can be expressed informally as follows: "Keep the window shut during the night." Notice that the requirement statement *references* the fluent Night of the TiP at the interface TiP!a, and *constrains* the fluent Shut of the window at the interface Win!d.

A possible specification for the security feature (SF) may be: "Fire `TiltIn` whenever `NightStarted` is observed (SF1). Once `TiltIn` is fired, do not fire `TiltOut` as long as `DayStarted` is not observed (SF2)." It should be noted that the specification is written only in terms of events at the interface between the machine and the world, while the requirement is written only in terms of the fluent properties of the problem world domains.

Climate Control Feature in Smart Home. The problem of the climate control feature (CCF) is shown in Figure 2. The requirement of this problem is: "Keep the window open when it is hot". Temperature Panel (TeP) fires an event `HotStarted` or `ColdStarted` to indicate the relationship between the preferred and the actual temperatures.

Correctness of Specifications. In addition to the three descriptions, the Problem Frames approach also provides a way of relating these descriptions

through the entailment relationship $W, S \models R$, showing how the specification, within a particularly context of the problem world, is sufficient to satisfy the requirement. This provides a template for structuring correctness proofs, and/or arguments for sufficiency/adequacy, of specifications [2].

An informal argument for adequacy of the specification may be provided as a positive event sequence: When the night starts, the time panel will generate the event `NightStarted` observed by the machine (TiP1). The security feature will fire `TiltIn` as soon as it observes `NightStarted` (SF1). `TiltIn` makes the window shut (Win1). Since the specification does not allow the window to tilt out until `DaytStarted` is observed (SF2), the window will remain shut during the night, thus satisfying the requirement (SR).

Although we have so far described the problem diagrams using an informal language, the Problem Frames approach is agnostic about the particular choice of the description language. We now give an overview of the description languge used in remainder of the paper.

2.2 The Event Calculus

The Event Calculus (EC) is a system of logical formalism, which draws from first-order predicate calculus. It can be used to represent actions, their deterministic and non-deterministic effects, concurrent actions and continuous change [10]. Therefore, it is suitable for describing and reasoning about event-based temporal systems such as the smart home application. Several variations of EC have been proposed, and the version we adopted here is based on the discussions in [11]. Some elementary predicates of the calculus and their respective meanings are given in Table 1.

Table 1. Elementary Predicates of the Event Calculus

Predicate	**Meaning**
Happens(a, t)	Action a occurs at time t
Initiates(a, f, t)	Fluent f starts to hold after action a at time t
Terminates(a, f, t)	Fluent f ceases to hold after action a at time t
HoldsAt(f, t)	Fluent f holds at time t
$t1 < t2$	Time point $t1$ is before time point $t2$

The Event Calculus also provides a set of domain-independent rules to reason about the system behaviour. These rules define how fluent values may change as a result of the events.

$$Clipped(t1, f, t2) \stackrel{\text{def}}{\equiv} \exists a, t[Happens(a, t) \wedge t1 \leq t < t2 \wedge Terminates(a, f, t)] \quad \text{(EC1)}$$

$$HoldsAt(f, t2) \leftarrow [Happens(a, t1) \wedge Initiates(a, f, t1) \wedge t1 < t2 \wedge \neg Clipped(t1, f, t2)] \quad \text{(EC2)}$$

For instance, the rule (EC1) states that Clipped(t1,f,t2) is a notational shorthand to say that the fluent f is terminated between times t1 and t2, whilst the rule (EC2) says that fluents that have been initiated by occurrence of an event continue to hold until occurrence of a terminating event. There are other such rules in the Event Calculus but we will omit them for space reasons. All variables in our formulae are universally quantified except where otherwise shown. We also assume linear time with non-negative integer values. We follow the rules of circumscription in formalizing commonsense knowledge [12], by assuming that all possible causes for for a fluent are given and our reasoning tool cannot find anything except those causes.

In the Event Calculus, given a requirement expressed using a `HoldsAt` formula, the abduction procedure will find all `Happens` literals via a system description and the Event Calculus meta-rules. For example, given the requirement `HoldsAt(Open, 4)`, and the domain rule `Initiates(TiltOut, Open, t)`, the meta-rule (EC2) allows us to abduce that `Happens(TiltOut, 3)` is a possible event squence to satisfy the requirement.

2.3 Relating Event Calculus to Problem Frames

As discussed in [9,15], requirements are largely about some desired properties in the problem world, and can be expressed using the `HoldsAt` predicate. Problem world domains are about causality between events and fluents: event-to-fluent causality can be expressed using the `Initiates` and `Terminates` predicates, and the fluent-to-event causality is expressed using the `HoldsAt` and `Happens` predicates. Specifications are about events generated by machines, and can be expressed using the `Happens` predicate. Event and fluent names will be parameterised with the names of domains that control them. For instance, we will write `TiltIn(SF)` to refer to the event `TiltIn` controlled by the domain Security Feature. The same event or fluent name with different parameters denote distinct events or fluents respectively.

Once described in this way, the $W, S \models R$ entailment of problem diagrams can be instantiated in the Event Calculus, allowing us to prove the correctness of a specification, with respect to a problem world context and a requirement [15].

3 Identifying Problem Interactions

This section describes the formal basis of our approach, while working through the running examples introduced in Section 2.

3.1 Abducing Failure Event Sequences in Problem Composition

Let n be the total number of problem diagrams that have been initially created in the workspace, where $1 \leq n$. Each problem diagram has the Event Calculus descriptions of the requirement, relevant problem world domains, and the specification. Furthermore, each specification is assumed to be correct with respect

to its problem world context and the requirement: for every diagram i in the workspace, where $1 \leq i \leq n$, the entailment $W_i, S_i \models R_i$ holds. Since problem diagrams are created incrementally, the n^{th} diagram is typically the newly added diagram.

Let W be $W_1 \wedge \cdots \wedge W_n$, denoting the finite conjunction of all `Initiates`, `Terminates`, and `Happens` formulae in the current workspace. It is a conjunction because each `Initiates` and `Terminates` formula, for instance, describes a rule by which a fluent changes its value, and if the rule can be applied in a subproblem, it should be possible to apply the same rule in the composed system. We assume that W is consistent, meaning for instance that there are no two rules in W that allow a fluent to be true and false at the same time.

Let S be $S_1 \wedge \cdots \wedge S_n$, denoting the finite conjunction of the `Happens` formulae in the specifications of problem diagrams in the workspace. Finally, let R be $R_1 \wedge \cdots \wedge R_n$, denoting the finite conjunction of `HoldsAt` formulae of the requirements in the problem diagrams. We consider conjunction, rather than disjunction, as the composition operator for any composed requirement, because any disjuncted requirement will be trivially satisfied if one of the individual requirements has been satisfied (which is the case).

Our focus, therefore, is on requirements: in particular, requirements that interaction. When the composed requirements is not always satisfiable, we would like to identify the event sequences where the composition may fail. Again, the possibility of composition failures does not necessarily mean the system is not useful: these failures may be tolerated, even desired in some cases, or prevented from arising by modifying the problem context. The composed system should have the property $W, S \models R$. The question raised then is: What are the possible failure event sequences in this system? In other words: What are the event sequences where this entailment may not hold?

Let Δ be the set of all possible event sequences (ordered Happens literals) permitted by the system, i.e. W and S. One way to check whether R can be failed is as follows. For every $\sigma \in \Delta$, verify whether the entailment (1) holds.

$$W, S, \sigma \models R \tag{1}$$

Let us denote the set of event sequences that satisfy the entailment (1) as Δ_1 and the set of event sequences that do not satisfy the entailment (1) as Δ_2. Clearly, $\Delta = \Delta_1 \cup \Delta_2$. There are two major limitations to finding Δ_2 through deduction. In a system with a reasonable number of events and fluents, verifying the relationship for a good length of time will require a large Δ, which is usually difficult to obtain. Secondly, this approach can be highly inefficient because it requires checking exhaustively [8].

In such circumstances, logical abduction is regarded as more efficient [8]. Logical abduction is a procedure that, given a description of a system, finds all event sequences that satisfy a requirement. Since an abduction procedure will return the event sequences Δ_1 that satisfy the goal, it is not possible to obtain Δ_2 using the abduction procedure on the entailment relation (1).

In order to identify failure event sequences, we take the negation of the composed requirement $\neg R$ as the requirement to be satisfied, whilst W and S serve

as the same description of the composed system. Given the uniqueness of fluent and event names, completion of `Initiates` and `Terminates` predicates, and the event calculus meta-rules, the procedure will find a complete set of event sequences Δ_2, such that the entailment (2) holds for every member ϵ of Δ_2 [12].

$$W, S, \epsilon \models \neg R \tag{2}$$

Since Δ_2 also is a set of event sequence permitted by the composed system, any ϵ is a member of Δ. Each ϵ is a failure event sequence, and is a refutation of (1). If Δ_2 is empty, Δ equals Δ_1, and all valid sequences of events permitted by the composed system will lead to the satisfaction of the requirement in (1).

Since our Event Calculus formulae are annotated with the names of problem world domains in problem diagrams, when Δ_2 is not empty, each ϵ will contain references to elements in the problem diagrams. This information allows us to relate the results of abduction procedure back to the corresponding problem diagrams in the workspace.

3.2 Smart Home Example

In order to illustrate a simple application of the approach, we will first formalise the requirements, specifications and the descriptions of the problem world domains discussed in Section 2. Natural language descriptions of all formulae given below are provided in Section 2.

Security Feature. The requirement for the security feature (SR), described in Section 2, can be formalised as follows.

$$HoldsAt(Night(TiP), t) \rightarrow HoldsAt(Shut(Win), t + 1) \tag{SR}$$

This formula is, in fact, stronger than the natural language statement. The formula says that at every moment that is night, the window should be shut at the next moment, requiring the window to be shut until one time unit after the night has passed. This formulation is chosen for its simplicity. The behaviour of the window domain in the security problem is given below.

$$Initiates(TiltIn(SF), Shut(Win), time) \tag{Win1}$$

$$Initiates(TiltOut(SF), Open(Win), time) \tag{Win2}$$

$$Terminates(TiltIn(SF), Open(Win), time) \tag{Win3}$$

$$Terminates(TiltOut(SF), Shut(Win), time) \tag{Win4}$$

$$HoldsAt(Open(Win), time) \leftrightarrow \neg HoldsAt(Shut(Win), time) \tag{Win5}$$

Parameterisation of the event and fluent names is important because (Win1), for instance, allows only the TiltOut event generated by the security feature

to affect the fluent Shut. The behaviour of the time panel domain is described below.

$$[HoldsAt(Day(TiP), time-1) \wedge HoldsAt(Night(TiP), time)] \leftrightarrow Happens(NightStarted(TiP), time) \quad \text{(TiP1)}$$

$$[HoldsAt(Night(TiP), time-1) \wedge HoldsAt(Day(TiP), time)] \leftrightarrow Happens(DayStarted(TiP), time) \quad \text{(TiP2)}$$

Finally, the specification of the security feature can be formalised as follows.

$$Happens(NightStarted(TiP), time) \rightarrow Happens(TiltIn(SF), time) \quad \text{(SF1)}$$

$$[Happens(NightStarted(TiP), time) \wedge \neg Happens(DayStarted(TiP), time1) \wedge time \leq time1] \rightarrow \neg Happens(TiltOut(SF), time1) \quad \text{(SF2)}$$

Climate Control Feature. Formalisation of the requirements, problem world domains and the specification of the climate control feature is given below. Since the behaviour of the window is the same in both problems, we will omit their formulae in this feature, but note that the TiltIn and TiltOut events will be parameterised with CCF, instead of SF.

$$HoldsAt(Hot(TeP), t) \rightarrow HoldsAt(Open(Win), t+1) \quad \text{(TR)}$$

$$[HoldsAt(Cold(TeP), time-1) \wedge HoldsAt(Hot(TeP), time)] \leftrightarrow Happens(HotStarted(TeP), time) \quad \text{(TeP1)}$$

$$[HoldsAt(Hot(TeP), time-1) \wedge HoldsAt(Cold(TeP), time)] \leftrightarrow Happens(ColdStarted(TeP), time) \quad \text{(TeP2)}$$

$$Happens(HotStarted(TeP), time) \rightarrow Happens(TiltOut(CCF), time) \quad \text{(TF1)}$$

$$Happens(HotStarted(TeP), time) \wedge [\neg Happens(ColdStarted(TeP), time1) \wedge time \leq time1] \rightarrow \neg Happens(TiltIn(CCF), time1) \quad \text{(TF2)}$$

Detecting Interactions. R in this case is TR $\wedge$ SR; W is the conjunction of (Win1–Win5), similar formulae for the window in the climate control problem, (TiP1), (TiP2), (TeP1) and (TeP2); and S is the conjunction of (SF1), (SF2), (TF1) and (TF2). In order to detect posssible failure event sequences in the composition of the two problems, we will first take the negation of the composed requirement, which can be stated as:

$$(HoldsAt(Night(TeP), t) \wedge HoldsAt(Open(Win), t+1)) \vee (HoldsAt(Hot(TeP), t) \wedge HoldsAt(Shut(Win), t+1)) \quad (\neg R)$$

The abduction procedure, in this case, will work as follows. It may be either day or night, and hot or cold, and the window may be open or shut at the beginning, and the procedure will condsider every valid combination. Suppose the window is initially open during a hot day at a symbolic time t1. In order to abduce event sequence for $HoldsAt(Hot(TeP), t1) \wedge HoldsAt(Shut(Win), t1+1)$, for instance, the procedure will look for events that can make the fluents hold.

Since it is already hot, $HoldsAt(Hot(TeP), t1)$ is true. In order to satisfy $HoldsAt(Shut(Win), t1+1)$, the procedure will look for event that can turn the current state $HoldsAt(Open(Win), t1)$ into $HoldsAt(Shut(Win), t1+1)$.

According to the domain rule (Win1), its counterpart for the climate control problem, and the Event Calculus meta-rule (EC2), if the event TiltIn(SF) or TiltIn(CCF) happens, the window will be shut at the next time point, provided the event TiltOut(SF) or TiltOut(CCF) does not happen at the same time. The event TiltIn(SF) will be fired when NightStarted(TeP) is fired, according to (SF1), and NightStarted(TeP) is triggered when the day turns into night, according to (TeP1). The event TiltOut(SF) will not happen at the same time because of (SF2). TiltOut(CCF) in (TF1) will not happen at the same time because it has been hot for a while.

In other words, in one possible failure event sequence, the window is open during a hot day, and the night soon begins. In that case, the window will be shut according to the specification of the security feature, because the climate control feature cannot prevent the window from being shut, thus resulting in a failure situation where the smart home is hot but the window is shut. This, of course, is a single event sequence and there may be other such event sequences, and the abduction procedure will find all of them in one pass. From the event sequence obtained, we know how the security problem and the climate control problem can interact.

The failure event sequence in this composition is due to the fact that no precendence between the security and the climate control requirements has been defined. It is, of course, possible that the smart home is situated in a world where it is never too hot at night. The interaction only arises under certain conditions, and the abduction procedure can find those conditions. (Strictly speaking, the abduction procedure cannot reason about the changes from day to night unless the events for these changes are given. That is due to the frame axiom in the Event Calculus. The examples we implemented in the next section include these events, but for space reasons we have ommitted them.)

A similar failure may arise if the specifications of individual problems are too strong. For example, another way to satisfy the security requirement is to have a specification that fires the TiltIn event at every time point, thus ensuring that the window is shut at all times. Similarly, the climate control requirement can be satisfied by another specification that fires the TiltOut event at every time point. They both satisfy the individual requirements, but any chance of composition is prevented because the specifications are too strong. Again, the same procedure can be used to identify those conditions in descriptions of problem diagrams. A detailed discussion, however, is beyond the scope of the paper.

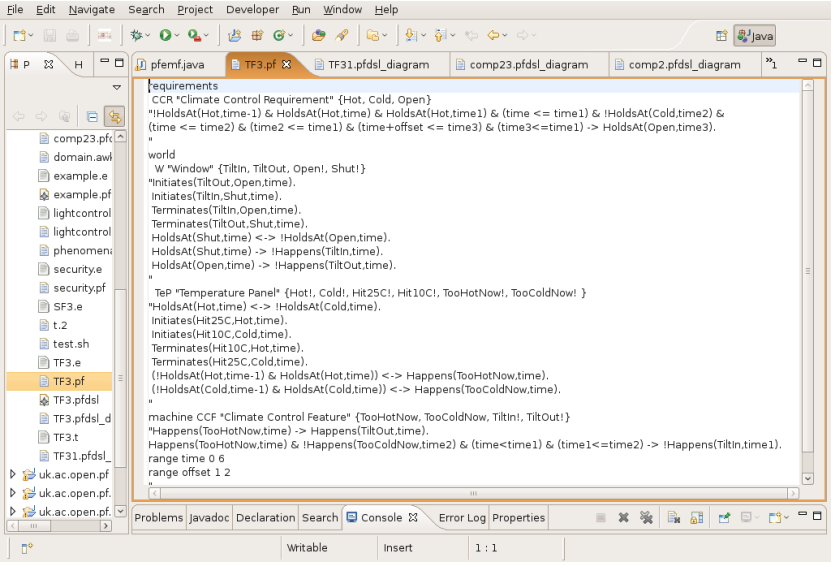


Fig. 3. An input file to create a problem diagram

Performing this abduction procedure manually is both labourious and error-prone. Fortunately, there are several implementation of this procedure for the Event Calculus. In the next section, we describe an end-to-end tool for detecting interacting problems that automates much of what has been discussed.

4 Detecting Interacting Problems Using the OpenPF

This section gives a brief overview of the OpenPF tool, with a particular emphasis on how interacting problems can be discovered early in the development.

4.1 Creating Problem Diagrams

An easy way to create problem diagrams using the OpenPF is through an input file that defines the names of the requirement, problem world domains and the machine, together with the Event Calculus formulae as their descriptions. Figure 3 shows an extract from the input file to create the climate control problem diagram and its descriptions.

Our OpenPF tool will check the syntax of the above input and the conformance of the Event Calculus formulae to the problem diagram. For instance, the tool will not allow two problem world domains to have the same names as otherwise there may be ambiguity in the produced EC formalae. Furthermore, it will check whether the requirement description refers to event names: since the requirements should be written in terms of fluents, such descriptions are not allowed. The tool will also annotate the event calculus formulae with the appropriate domain names: for instance, the fluent term Open will be written as

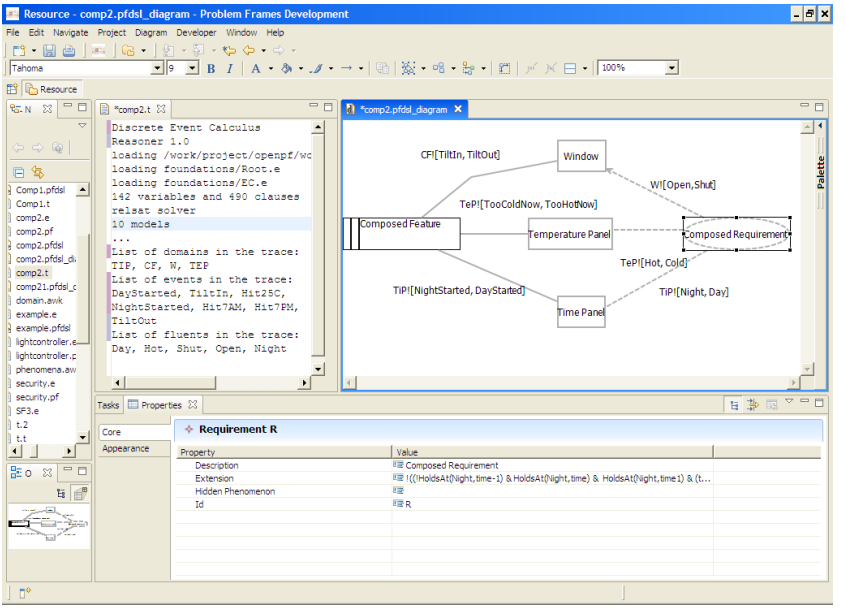


Fig. 4. Generated problem diagram with the Event Calculus descriptions

Open(W) because the window domain assigns values to the fluent. It will also generate a problem diagram from the input.

4.2 Detecting Interactions in the Running Example

Once a problem diagram is created, the OpenPF tool can generate a composition diagram such as the one shown in Figure 4 for the running example. When the diagram is created, the tool automatically generates the Event Calculus script to abduce failure event sequences for the composition. The script will include the conjunction of all formulae for problem world domains in all individual problem diagrams, the conjunction of all formulae for the specifications, and the negation of the conjunction of the requirements formulae (as shown in the property window in Figure 4).

The Event Calculus script is then fed to the off-the-shelf abductive reasoner, `Decreasoner` [12,13], in order to abduce the event sequences satisfying the negated requirement. `Decreasoner` will translate the abduction problem in the Event Calculus into a SAT problem and solve it using the solver `Relsat` [16,17]. SAT results are then translated back into the Event Calculus literals by `Decreasoner`. From the output of `Decreasoner`, the OpenPF tool will capture the abduction output, and relate it to the elements in the problem diagrams. In one view, the tool can show the event sequences of the interactions, and in another view, it will pinpoint the list of problem world domains, events and fluents involved in a the interaction (as shown in the panel to the left of

the diagram in Figure 4). Once an input file as shown in Figure 3 is provided, the rest of the tasks of finding interacting problem diagrams, or even individual elements with a diagram, is done automatically.

Our initial evaluation criterion is to implement the idea that identifying possible failure event sequences in problem composition can be done efficiently through logical abduction. Our implementation of the OpenPF using Problem Frames, Event Calculus, decreasoner, and Model-driven Eclipse plugins, and the smart home examples have demonstrated the viability of our idea. An abduction example involving 140 variables and 440 Event Calculus clauses has been computed in less than one second on a standard personal laptop. Since the abduction procedure and modern SAT solvers such as Relsat are efficient, it gives us confidence that a framework such as the OpenPF will scale when applied to larger examples.

5 Related Work

Russo *et al.* [8,18] provide theoretical insights on the use of abductive reasoning in the analysis of requirements. Given a system description and an invariant, Russo *et al.* propose using of an abduction procedure to generate a complete set of counterexamples, if there is any, to the invariant. Rather than analyse explicit safety properties, we use logical abduction to identify possible interactions between problems, with the assumption that conjunction will be the eventual composition operator. Failure event sequences suggested by our approach may not be sound (if the interactions can be tolerated), but they provide an early indication of possible failures in composition.

Wang *et al.* [5] propose a SAT-based framework for monitoring run-time satisfaction of requirements and diagnosing the problems when errors happens. In order to diagnose the components where errors originate, the framework logs the system execution and when goals are not satisfied, the traces are preprocessed and transformed into a SAT problem using propositional logic. Although their aim and ours are similar, we work with early requirements models where there is no running system to monitor. Moreover, our approach generates possible failure event sequences by abduction procedure.

van Lamsweerde et al [19] propose a framework for detecting requirements inconsistencies at the goal level and resolving them systematically. Unlike the KAOS approach [19], we reason about the system behaviour in a bottom-up fashion. Once the behaviour of individual solutions are specified, failure event sequences can be generated automatically. A way of resolving the composition problem such as precedence are discussed in [9].

Nentwich *et al* [20] describe a tool-supported approach for describing consistency constraints and checking them across various resources. Similarly, Egyed [21] presented a tool for instantly checking consistency of UML diagrams. Although inconsistency checking also plays an important role in our approach, we are detecting run-time interactions rather than static inconsistency in the representation of artefacts.

Seater and Jackson [22] propose a systematic way to derive specifications using the Problem Frames approach, and they use the Alloy language and Alloy Analyzer to check the validity of the derivation. Our work is complementary in the sense that they are concerned with decomposing and specifying individual problems, but we are concerned with the composition of the individual problems.

6 Conclusions and Future Work

In this paper, we examined the issue of problem interactions: these are situations where the conjunction of solutions to smaller problems introduce new, often unexpected, problems in the composed solution. Although checking whether some given problems can be composed is relatively easy, identifying situations where the composition may fail can be difficult. In this paper, we proposed that identification of problem interactions where composition cannot be achieved fully can be done through logical abduction. We have used the OpenPF tool to demonstrate our idea using examples taken from a smart home application.

The issue of identifying possible failure event sequences is closely related to the question of suggesting viable corrective actions to resolve undesired interactions. We are currently investigating how requirements engineers can use the early feedback obtained through logical deduction in order to come up with proposals for corrective actions.

Although, we focused on problem solving approaches that defer the concerns of composition, other problem solving approaches in requirements engineering may have a similar issue. For instance, when there is a need to modify a goal tree, or to merge smaller goal trees, the question of finding event sequences leading to possible failures may be raised. Therefore, we conjecture that our approach can be applied, with little or no modification, in those cases. We are also investigating in this direction.

Acknowledgements. We would like to thank our colleagues at the Open University, in particular, Michael Jackson, and the anonymous reviewers for their helpful comments and suggestions. This research is funded by the EPSRC, UK.

References

1. Parnas, D.L., Lawford, M.: The role of inspection in software quality assurance. IEEE Trans. Softw. Eng. 29(8), 674–676 (2003)
2. Jackson, M.: Problem Frames: Analyzing and structuring software development problems. ACM Press & Addison Wesley (2001)
3. Robinson, W.N., Pawlowski, S.D., Volkov, V.: Requirements interaction management. ACM Computing Surveys 35(2), 132–190 (2003)
4. Jureta, I., Mylopoulos, J., Faulkner, S.: Revisiting the core ontology and problem in requirements engineering. In: Proceedings of the 2008 16th IEEE International Requirements Engineering Conference, pp. 71–80. IEEE Computer Society Press, Los Alamitos (2008)

5. Wang, Y., McIlraith, S.A., Yu, Y., Mylopoulos, J.: An automated approach to monitoring and diagnosing requirements. In: Proceedings of the International Conference on Automated Software Engineering, pp. 293–302. ACM, New York (2007)
6. Shanahan, M.: Prediction is deduction but explanation is abduction. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 1055–1060. Morgan Kaufmann, San Francisco (1989)
7. Denecker, M., Schreye, D.D.: Sldnfa: an abductive procedure for normal abductive programs. In: Proc. of the International Joint Conference and Symposium on Logic Programming, pp. 686–700. MIT Press, Cambridge (1992)
8. Russo, A., Miller, R., Nuseibeh, B., Kramer, J.: An abductive approach for analysing event-based requirements specifications. In: Stuckey, P.J. (ed.) ICLP 2002. LNCS, vol. 2401, pp. 22–37. Springer, Heidelberg (2002)
9. Laney, R., Tun, T.T., Jackson, M., Nuseibeh, B.: Composing features by managing inconsistent requirements. In: Proceedings of 9th International Conference on Feature Interactions in Software and Communication Systems (ICFI 2007), pp. 141–156 (2007)
10. Shanahan, M.P.: The event calculus explained. In: Woolridge, M.J., Veloso, M. (eds.) Artificial Intelligence Today. LNCS, vol. 1600, pp. 409–430. Springer, Heidelberg (1999)
11. Miller, R., Shanahan, M.: The event calculus in classical logic - alternative axiomatisations. Journal of Electronic Transactions on Artificial Intelligence (1999)
12. Mueller, E.T.: Commonsense Reasoning. Morgan Kaufmann, San Francisco (2006)
13. Decreasoner, http://decreasoner.sourceforge.net/
14. Kolberg, M., Magill, E., Marples, D., Tsang, S.: Feature interactions in services for internet personal appliances. In: Proceedings of IEEE International Conference on Communications (ICC 2002), vol. 4, pp. 2613–2618 (2001)
15. Classen, A., Laney, R., Tun, T.T., Heymans, P., Hubaux, A.: Using the event calculus to reason about problem diagrams. In: Proceedings of International Workshop on Applications and Advances of Problem Frames, pp. 74–77. ACM, New York (2008)
16. Bayardo Jr., R.J., Schrag, R.: Using CSP look-back techniques to solve real-world SAT instances. In: AAAI/IAAI, pp. 203–208 (1997)
17. Relsat, http://code.google.com/p/relsat/
18. Russo, A., Nuseibeh, B.: On the use of logical abduction in software engineering. In: Chang, S.K. (ed.) Software Engineering and Knowledge Engineering. World Scientific, Singapore (2000)
19. Lamsweerde, A.v., Letier, E., Darimont, R.: Managing conflicts in goal-driven requirements engineering. IEEE Trans. Softw. Eng. 24(11), 908–926 (1998), http://dx.doi.org/10.1109/32.730542
20. Nentwich, C., Capra, L., Emmerich, W., Finkelstein, A.: xlinkit: a consistency checking and smart link generation service. ACM Trans. Interet Technol. 2(2), 151–185 (2002)
21. Egyed, A.: Instant consistency checking for the uml. In: Proceedings of the International Conference on Software Engineering, pp. 381–390. ACM Press, New York (2006)
22. Seater, R., Jackson, D.: Requirement progression in problem frames applied to a proton therapy system. In: Proceedings of RE 2006, Washington, DC, USA, pp. 166–175. IEEE Computer Society Press, Los Alamitos (2006)

Composing Models for Detecting Inconsistencies: A Requirements Engineering Perspective

Gilles Perrouin[1], Erwan Brottier[2], Benoit Baudry[1], and Yves Le Traon[3]

[1] Triskell Team IRISA/INRIA Rennes Campus de Beaulieu, 35042 Rennes, France
{gperroui,bbaudry}@irisa.fr
[2] France Télécom R&D, 2 av. Pierre Marzin, 22 307 Lannion Cedex, France
erwan.brottier@orange-ftgroup.com
[3] ENST Bretagne, 2 rue de la Châtaigneraie, CS 17607,
35576 Cesson Sévigné Cedex France
Yves.letraon@telecom-bretagne.eu

Abstract. [**Context and motivation**] Ever-growing systems' complexity and novel requirements engineering approaches such as reuse or globalization imply that requirements are produced by different stakeholders and written in possibly different languages. [**Question/ problem**] In this context, checking consistency so that requirements specifications are amenable to formal analysis is a challenge. Current techniques either fail to consider the requirement set as a whole, missing certain inconsistency types or are unable to take heterogeneous (i.e. expressed in different languages) specifications into account. [**Principal ideas/ results**] We propose to use model composition to address this problem in a staged approach. First, heterogeneous requirements are translated in model fragments which are instances of a common metamodel. Then, these fragments are merged in one unique model. On such a model inconsistencies such as under-specifications can be incrementally detected and formal analysis is made possible. Our approach is fully supported by our model composition framework. [**Contribution**] We propose model composition as means to address flexibility needs in requirements integration. Threats to validity such as the impact of new requirements languages needs to be addressed in future work.

Keywords: model-driven requirements engineering, flexible inconsistency management, model composition.

1 Introduction

Cheng and Atlee [1] have reviewed state of the art of current requirements engineering research and identified future directions. Amongst those, two seem particularly relevant to address ever-growing system complexity, shorten engineering time and maximize value: *Globalization* and *Reuse*. Globalization suggest to engineer systems in geographically distributed teams in order to benefit from a continuous working force (24h/day), close distance of customers and resource optimization. Reuse offers to capitalize on requirements value by wisely

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 89–103, 2009.

re-applying the same requirements in a product-line context. These promising research directions have a strong impact on the definition of the requirements themselves. First, requirements of a single system will be handled by several engineering teams having different habits and therefore inducing communication challenges [1]. Second, reusing requirements in a new context may imply having to deal with different formalisms used for their description. As mentioned by Sommerville [2], a Software Requirements Specification (SRS) is captured by a collection of viewpoints, described by system authorities (stakeholders, existing system documentation, and so on). Viewpoints encapsulate partial requirements information, described by heterogeneous models i.e. expressed in various languages (depending on stakeholders preferences and skills) and relating to different crosscutting concerns [3].

Globalization and reuse also represent a challenge for consistency management. Indeed, models forming viewpoints are likely to be inconsistent due to the amount of heterogeneous information involved and the number of stakeholders responsible of their productions. Requirements analysts need to detect inconsistencies among these models to reveal conceptual disagreements and drive the requirements elicitation process [4]. To do so, they need a global view of inconsistencies in order to decide whether they will tolerate inconsistency presence or not [5]. Model comparison techniques [4,6,7,8] have been proposed to detect logical contradictions with respect to consistency rules. These techniques are relevant to find static and inter-model inconsistencies. But they have some important limitations. First, consistency rules in a multi-formalism context must be written for each possible pair of languages. Second, these techniques are inefficient to provide a measure of the overall SRS consistency since consistency rules checks models two by two. As a result, they are not suitable to detect under-specifications (a lack of information detected between two models may be resolved by a third one). Moreover, dynamic inconsistencies are undetectable because formal techniques enabling their detections can not be used as they require a global model of the SRS.

Composing models can help to overcome these limitations by providing one global model from a set of models providing an unified view [9] of the requirements with respect to a particular purpose (e.g. functional requirement simulation). Regarding inconsistencies detection, model composition translates the inter-model consistency problem into an intra-model consistency one. This has numerous advantages. First, consistency rules can be defined on one unique metamodel and hence are much easier to specify. Second, dynamic inconsistencies can be readily checked by formal tools.

However, current composition techniques [9,10,11,12,13,14] do not fully address our problem. Indeed, they do not support the composition of heterogeneous models since they compose models in the same formalism. Most of these approaches assume that models are conforming to their metamodel prior to their composition which is not common place in requirements engineering practice [15]. Finally, they do not emphasize of ensuring traceability during composition which is required to determine inconsistency source, such as stakeholders conflicts.

Building on our experience on model composition [13], we propose in this paper a generic composition process addressing the above issues. First, we extract information from heterogeneous models and translate it in terms of a set of model fragments. This step is called interpretation. The second step, called fusion, builds a global model by composing model fragments. The resulting global model can be analyzed with respect to under-specifications and other inconsistency types through dedicated diagnostic rules potentially covering the SRS as a whole. By postponing inconsistency detection in the global model it is possible to resolve inconsistencies in an order only determined by the requirements engineers. Therefore, we provide them the flexibility required to drive inconsistency management. We also automatically compute traceability links between elements of input models and the global one. This is useful to trace inconsistencies back in models, providing valuable feedback to stakeholders. This process is fully supported by a model-driven platform [13,16], integrated into the Eclipse framework. Section 2 outlines our model composition process. Section 3 details the fusion step and illustrates how inconsistent models can be composed. Section 4 illustrates how various kinds of inconsistencies can be detected. Section 5 highlights some relevant work. Section 6 concludes the paper and sketches some interesting perspectives.

2 Process Overview

In this section, we describe our composition-and-check process. It is a result of researches carried out in the context of the R2A[1] project (R2A stands for Requirement To Analysis) project, initiated in collaboration with THALES and FRANCE TELECOM. Its goal is the definition of a framework for analyzing requirements, simulating functional requirements [16] and generating software artifacts from them [17]. This process is completely based on a model-driven approach and consists of three sequential steps, as shown in Figure 1. It starts from a set of input models, expressed in various *input requirements languages* (IRL). These models are composed during the *interpretation* and *fusion* steps, presented in section 2.1. These steps result in a *global model* and a *traceability model*. These two models are the inputs of a *static analysis* step which consists in checking the global model according to consistency rules and producing a consistency verdict. This last step is described in section 2.2.

2.1 Composition

Input models to compose are produced by stakeholders. They capture specific parts of the software requirements specification (SRS) and describe partial information of the requirements. They may be inconsistent and can be described with various input requirements languages as depicted in Figure 1 (a preliminary parsing step presented in [13] is required for textual specifications.). Furthermore, they describe different concerns of the system-to-be. Composing such

[1] http://www.irisa.fr/triskell/Softwares/protos/r2a/r2a_core?set_language=en

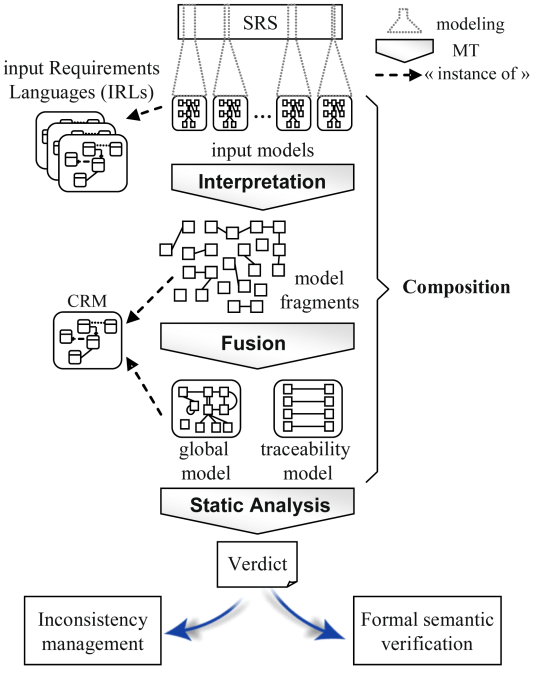


Fig. 1. Overview of the composition-and-check process

input models requires to state precisely the following: 1) Which information must be extracted from input models, 2) How this information is obtained from these input models and 3) How extracted pieces of information must be combined to obtain one global model.

Core Requirements Metamodel. The *Core Requirements Metamodel* (CRM) defines information that will be extracted from input models and composed. Two main factors influence the definition of the CRM. The first one is related to the answer that we will give to the first point i.e. the elicitation of a subset of the IRLs' concepts on which we will base analysis on the global model. This elicitation is the result of a negotiation between stakeholders to determine what are the most important concepts according to their respective viewpoints.

The second important factor is the type of analysis that is targeted by the check process. If dynamic inconsistency checking is required, a formal operational semantics of the CRM has to be given.

Interpretation. The first step of the process, called *interpretation*, addresses the second point. This step is detailed in previous work [13] and is beyond the scope of this paper. Basically, the interpretation extracts relevant information in input models and translates it in terms of *model fragments*, which are instances of the CRM. The interpretation is governed by a set of interpretation rules matching IRLs' concepts (with respect to optional guards) and producing corresponding

model fragments instances of the CRM. These interpretation rules have been defined in collaboration with THALES' requirements analysts in order to validate "correctness by construction" of interpretation rules.

Fusion. The second step of the process called *fusion* addresses the third point. From a model-driven perspective, the fusion step is supported via a model composition technique which provides a high-level of flexiblity (see Section 3.3 for details). As for interpretation, fusion is described by a set of *fusion rules*. The fusion consists in detecting and resolving *overlaps* between model fragments. An overlap is a situation where two sets of related model elements in different models are semantically equivalent i.e. designate common features of the domain of discourse [18]. Overlap resolution aims at producing a compact representation of information captured by interpretation, i.e. the global model. Without replacing a complete semantic analysis, the fusion makes explicit semantic links between input models. This step is detailed and illustrated in section 3.

Interpretation and fusion rules automatically build traceability information necessary to identify elements which are the cause of an inconsistency.

2.2 Static Analysis

The third step of the process, called static analysis, is performed once the global model is available. Two kinds of consistency rules are checked. The first ones, called structural inconsistencies, check if the global model fulfills at least the CRM constraints expressed with MOF (cardinalities, composition...). Two kinds of inconsistencies can be detected at this stage:

- *Under-specification* is detected if one property value of one object has fewer elements than specified by the property cardinality. It means that information is missing in input models and the global model is incomplete with regards to the targeted formal analysis technique,
- *Logical contradiction* is detected if one property value of one object has more elements than specified by the property cardinality. It means that at least two input models overlap semantically but this overlap is inconsistent.

The second kind of consistency rules, called *static semantics inconsistencies*, is complex and is generally described with OCL rules. Intuitively, these rules correspond to well-formedness rules defining business-specific restrictions on metamodels. However these rules can be difficult to write in OCL for stakeholders who do not have a technical expertise. To help such stakeholders, we propose the notion of *diagnostic rules* which are easier to write for stakeholders and enable to provide meaningful information in a non-technical form when a rule is violated. The composition-and-check process can be used iteratively in order to limit the amount of managed information and the number of inconsistencies to solve. As the fusion step can take as an input the global model (it is just a large model fragment). It is then possible to check how a new input model impacts a consistent global model. When no inconsistencies are detected, the targeted formal verification technique can be applied. When input models

(and corresponding parts of the SRS) are updated to take into account results of the formal verification, a new cycle can be started. These points will be detailed in section 4.2.

3 Combining Inconsistent Models via Fusion

3.1 Running Example: The RM

The RM metamodel is the current CRM of the R2A platform. It captures a functional and a data description of the system. It is meant to capture functional requirements and the control flow between them. RM also allows for requirements simulation and system test cases generation within R2A platform [17]. Figure 2 illustrates an excerpt of the RM metamodel. It describes a state-based formalism where system actions are described as use cases (metaclass UseCase), enhanced with their activation conditions and effects (relationships PreCondition and PostCondition expressed as first order logic expressions).

Only few concepts defining these expressions are showed in Figure 2 (the Expression subclasses). Expressions contain references to property values of the system, represented by the metaclass PropertyValue. The OCL constraint 1 in Figure 2 ensures that referenced property values in any expression exist in

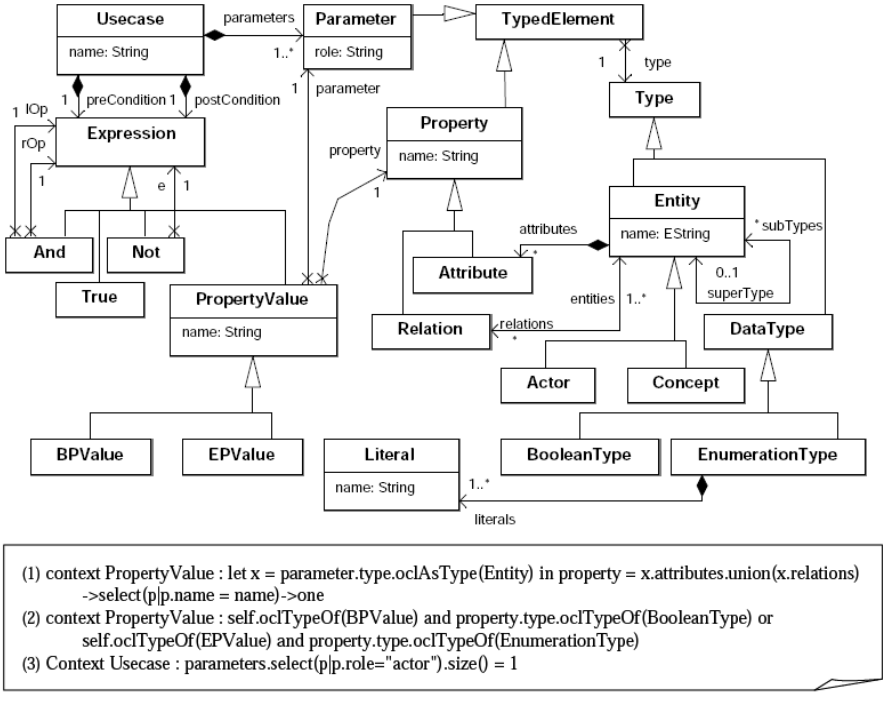


Fig. 2. A part of the RM metamodel used in R2A platform

the model (constraint 2 checks their type conformance). Use case contains a set of formal parameters (metaclass PARAMETER). Each parameter represents an ENTITY involved in the use case, which plays a particular ROLE (for instance "actor"). Only the actor parameter of a use case can trigger it and a use case has only one actor, as expressed by the OCL constraint 3.

Other notions in Figure 2 describe basically an entity-relationship diagram. Entities are either business concepts (CONCEPT) or actors (ACTOR) of the system (entity which can trigger at least one use case). Entities have properties which represent their possible states (ATTRIBUTE) and relationship with others (RELATION). Properties have a type (DATATYPE) which can be a BOOLEANTYPE or an ENUMERATIONTYPE (a finite set of literal values, representing strings, integers or intervals). Instances of attributes and relations are property values of the system. A system state is a set of entity instances and property values, corresponding to a system configuration at a given moment. The reader can refer to [13,15] for more details on this metamodel.

3.2 Dealing with Inconsistent Fragments

Figure 3 presents a set of RM model fragments, obtained by interpreting a set of input models (interpretation step). These input models represent the specification of a library management system (see [13] for details). Input models are written in RDL (Requirement Description Language [13]) which is a constrained form of natural English. For example, one sentence of library management system, interpreted in terms of model fragments of Figure 3 is:

```
The ‘‘book’’ must be registered before the ‘‘customer’’ can
‘‘borrow’’ the ‘‘book’’
```

Each fragment handles a piece of information, extracted from one input model by one interpretation rule as explained in section 2.1. The fragment (a) declares

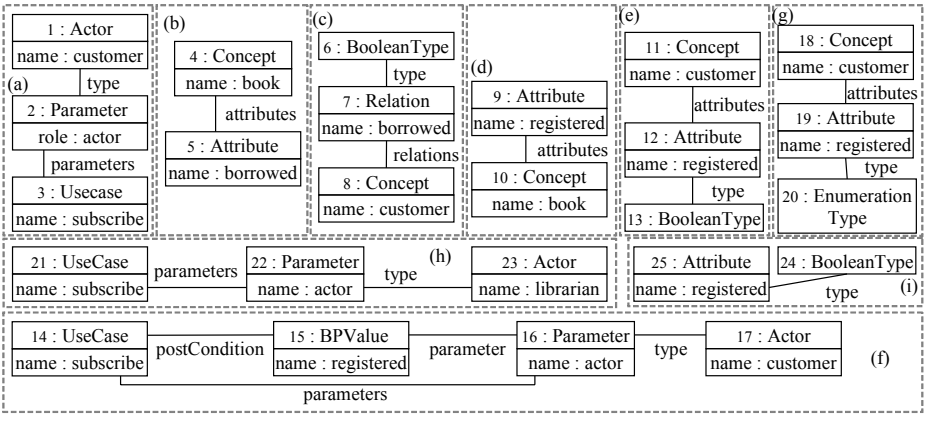


Fig. 3. Examples of RM model fragments

a use case subscribe which can be triggered by an actor *customer*. The fragment (b) states the existence of a business concept *book* which has a state *borrowed*, without information on its type. To handle such inconsistent fragments in a model (prior and after their fusion), it is necessary to allow non-compliance with respect to basic well-formedness rules of the metamodel such as multiplicities or types. We introduce the notion of a *relaxed metamodel* to cope with this situation.

Definition 1. *A relaxed CRM is a metamodel where the following properties hold:*

- *All metaclasses are considered as concrete (i.e. instantiable),*
- *All multiplicities between metaclasses are considered as '*',*
- *There is no containment relationship (treated as regular associations).*

Thanks to this notion it is possible to interpret partial and inconsistent model elements and to combine them. As mentioned in Section 2, inconsistency checking is progressively performed on the global model. Finally the global model is made consistent with respect to the original (un-relaxed) CRM. This process will be detailed in Section 4.

3.3 Fusion Principles

One simple approach to perform model fusion is to compare objects two by two and combine them when with respect to syntactical equivalence. Yet, this raises two issues (i) *designation clashes* and (ii) *type clash*. Designation clash [19] occurs when a single syntactic name in the requirements specification designates different real-world concepts (e.g. the pair (9,12) refers to different concepts while having the same name). A type clash arises when two different types ar given for the same concept (e.g. the pair (1, 8) refers to a customer in two different ways). It can be seen as specialization of a *terminology clash* [19]. To alleviate these issues, we need to let requirements analysts define fine-grained rules specializing fusion and resolve these clashes. To this aim, we propose two kinds of fusion rules (FR): *equivalence rules* (ER) and *normalization rules* (NR).

Rules of the first type define *equivalence ranges* and describe how to resolve them. An equivalence range is a set of objects which are considered as equivalent. Resolution targets the replacement of equivalent objects by one new object, where properties have been set properly to keep the whole information. Normalization rules aim at transforming the model after ER executions so that a violation of conformity reflects an inconsistency. Figures 4 and 5 give examples of FR specifications for the RM metamodel presented in Figure 2.

An ER is defined as an equivalence range constructor and an equivalence range resolution (constructor and resolution in the remainder). The constructor is a boolean expression which takes as inputs a pair of objects and returns true if they are equivalent. It aims at defining a set of equivalence ranges in the context of its owning rule. The resolution is used to replace all objects in

ER1 (a_1, a_2 : ATTRIBUTE): ATTRIBUTE
- $a_1.name = a_2.name$
  $\wedge\ a_1.attributes^{-1} = a_2.attributes^{-1}$.

ER2 (r_1, r_2 : RELATION): RELATION
- $r_1.name = r_2.name$.

ER3 (r : RELATION, a : ATTRIBUTE): RELATION
- a.*name* = r.*name*.
- *linkedEntities* = equRanges.collect(r : RELATION | r.linkedEntities) $\cup$ c.collect(a : ATTRIBUTE | a.owningEntity).

ER4 (u1, u2 : USECASE): USECASE
- u1.*name* = u2.*name*.

ER5 (p_1, p_2 : PARAMETER): PARAMETER.
- $p_1.name = p_2.name$.

ER6 (a_1, a_2 : ACTOR): ACTOR.
- $a_1.name = a_2.name$.

ER7 (c : CONCEPT, a : ACTOR): ACTOR
- c.*name* = a.*name*.

ER8 (c_1, c_2 : CONCEPT): CONCEPT.
- $c_1.name = c_2.name$.

ER9 (b_1, b_2 : BOOLEANTYPE): BOOLEANTYPE.
- $b_1.type^{-1} = b_2.type^{-1}$.

Fig. 4. Examples of equivalence rules

the equivalence ranges by a new object which captures their semantics. ERs are expressed with three ordered elements, listed in Figure 4: a *signature* (first line), a *constructor* boolean expression (the first element of the list) and a set of *resolution directives* (the second element being optional and used only in ER3). The signature handles the name of the rule, a type filter on the pair of objects checked by the constructor and the type of the object created by the resolution (*return type*). ER1 specifies for instance that objects of type ATTRIBUTE which have the same name and are related to the same ENTITY ($attributes^{-1}$ points out the opposite reference of *attributes*) are part of the same equivalence range. This ER illustrates how context can be part of overlaps identification. The resolution of such an equivalence range will produce an instance of ATTRIBUTE, as specified by its return type.

A resolution directive describes how to set the value of a particular property for the object created by the resolution. If no resolution directive is defined, the default policy is applied. As defined by Sabetzadeh et al. [9], it consists in making the union of the property value of each object in the equivalence range (with

NR1 : Execution of an imperative form of the constraint 1 in Figure 2 for each PROPERTYVALUE.
NR2 : Creation of an And tree with elements of u.preCondition for each USECASE.
NR3 (uc : USECASE) ? : card(uc.preCondition) = 0;
! : uc.preCondition := TRUE.new;

Fig. 5. Examples of normalization rules

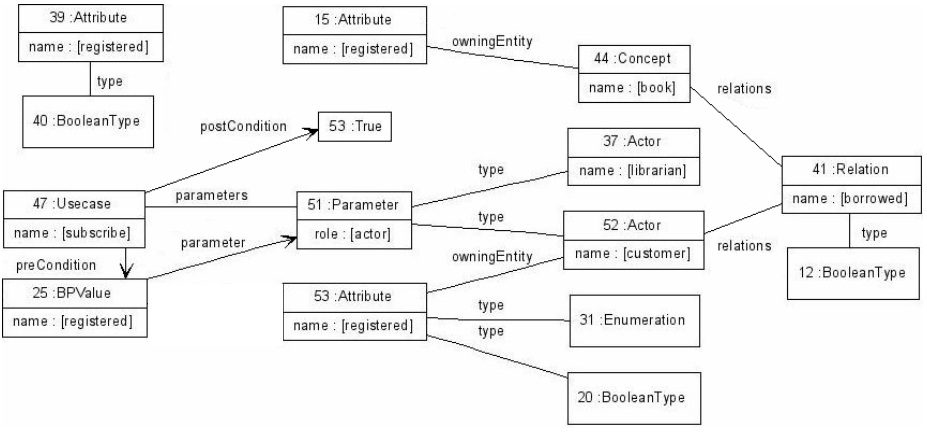


Fig. 6. Result of the application of Fusion rules on model fragments (Figure 3)

basic equality resolution for primitive types such as string or integer). Yet, some property values cannot be resolved by union (for instance the kind of a UML class representing one abstract class and one interface). In such cases, resolution directives are useful to resolve these overlaps. As an example, each property value of ATTRIBUTE instances created by the ER4 resolution is evaluated by the default policy as opposite to the *linkedEntities* property values of RELATION instances created by the ER3 resolution. The fusion of model fragments may not be performed in one step (identification of all equivalence ranges and resolution). Indeed, an object can be contained by more than one equivalence range. As an example, the object 1 is also contained by (1, 17) ER6. The stopping criterion of the fusion is satisfied when no more equivalence ranges have been identified. Some inconsistencies in a model do not reveal inconsistencies in the information captured but only

```
0   Algo Fusion(l1 : List<Object>, l2 : List<ER>) is
1   var resolved : List
2   do
3    resolved.clear()
4    forAll o1 in l1
5     forAll o2 in l1
6      forAll er in l2
7       if (er.constructor(o1, o2)) then er.equivalences.add(o1, o2)
8    forAll er in l2
9     forAll r in f.equivalenceRanges
10     var o : Object := er.resolution(r)
11     l1.add(o)
12     resolved.addAll(eq)
13   l1.elicit(resolved)
14  until(resolved.isVoid())
```

Fig. 7. Fusion Algorithm (equivalence rules application)

irrelevant ways to express this information. For instance, an instance of Usecase in a RM model must have one pre-condition exactly. If this use case has no condition of activation, its pre-condition must be an instance of the metaclass TRUE. This mapping to TRUE must be done if no partial specifications describe information about this pre-condition. NR3 is specified for this purpose.

The main part of the fusion algorithm is given in Figure 7. It processes a set of objects (from model fragments to merge) and a set of ER. It iterates until no more equivalence ranges have been resolved (or identified). The loop contains two steps: equivalence range identifications (lines 04-07) and their resolutions (lines 08-13). The method elicit(l:list of objects) removes all objects passed as parameter (it also deletes links between them and remaining objects). It is used to remove objects contained by a resolved ER. Figure 6 gives the final model obtained by executing this algorithm on model fragments of Figure 3 according to fusion rules of Figure 4 and 5. It is an instance of the relaxed CRM.

3.4 Traceability Computation

The composition process generates automatically a traceability model as introduced in section 2.1. This model stores the history of all kinds of rules (interpretation, fusion and normalization) that have been executed during the composition process. Each rule execution is associated to model elements that have been matched and produced. Model elements pertain either to input models, interpreted and composed model fragments or to the global model. Given such a traceability model, it is possible to compute a connected graph where nodes are model elements involved in at least one rule and vertices relates elements matched and produced for each rule.

4 Inconsistency Detection

We illustrate in this section our inconsistency detection process, introduced in section 2.2. This process is composed of two activities performed in parallel: *structural inconsistency detection* and *static semantics inconsistency detection.*

4.1 Structural Inconsistency Detection

Structural inconsistency detection checks conformance of the model with respect to the CRM definition expressible through MOF. For example, one constraint on the RM metamodel requires that any USECASE has at least one PARAMETER. Structural inconsistencies comprise logical contradictions: attribute *registered* of customer (object 53 in Figure 6) has two types which is clearly a non-sense. Concerning under-specifications, attribute *registered* (object 15) of *book* has no type at all. Structural inconsistencies for a given CRM are automatically detected by MOF-compliant tools. Therefore there is no need to define them for a new CRM. When no more structural inconsistency is detected, the global model can be cast to an instance of the original (un-relaxed) CRM.

4.2 Static Semantics Inconsistency Detection

We distinguish two categories of static semantics inconsistencies depending on the stakeholders who specify them. *Well-formedness* inconsistencies are related to rules domain experts define on the CRM (e.g. constraint 1 for the RM metamodel in Figure 2). *Custom* inconsistencies are violation of rules defined by requirements analysts. Enforcing such rules may be required to enable further formal analysis on the model. As mentioned in section 3, fusion outputs the global model as an instance of the relaxed CRM. Since for flexibility reasons [5] we do not want to impose a sequence in inconsistency detection activities, we cannot assume that the global model can be cast to an instance of the CRM. We therefore need to define well-formedness and custom rules on the relaxed CRM. While OCL is often used to specify static semantics rules, defining navigation paths across scattered elements can be tricky and rule violation feedback useless for stakeholders.

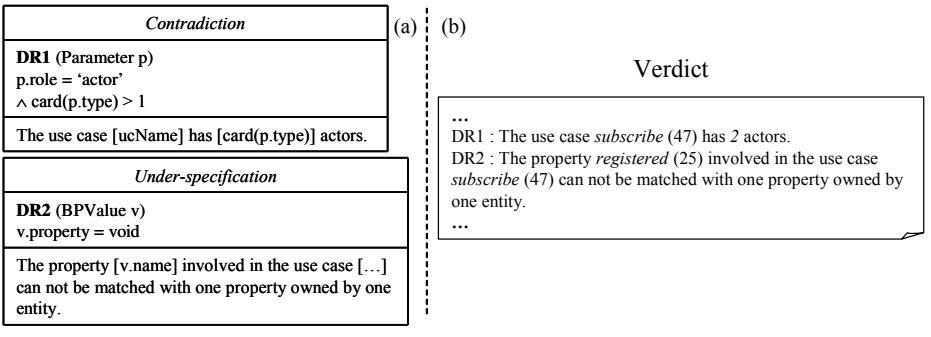


Fig. 8. Some DRs for the metamodel RM and an excerpt of the verdict

We offer the possibility to requirements analysts to define diagnostics rules (DR), defined for a given version of the relaxed CRM, to solve these problems. Diagnostics rules encapsulate a guard and a textual template. The guard is based on pattern matching as for interpretation and fusion rules. Hence, it is no longer required to specify navigation through the model to retrieve model elements. The template is a natural language sentence parameterized with expressions referring matched elements and navigating the global model. When a DR matches, an inconsistency is detected, the template is instantiated and added to a verdict log which can be reviewed by non-experts. Traceability links can be then navigated until pointing out inconsistent requirements in input models and rules involved in the production of elements responsible of the inconsistency. A few examples based on the RM metamodel are provided in Figure 8a. For instance, DR1 declares that there is a contradiction if a use case has more than one actor. Indeed, this DR is related to the OCL constraint 3 defined Figure 2). As the verdict illustrates (Figure 8b), the feedback for this rule is much more understandable for non-experts.

When both static semantics and structural inconsistencies are resolved we cast the global model to an instance of the original CRM which is fully consistent and amenable to further analysis.

While having no inconsistency detection sequence is a good point for flexibility, it can be disturbing for requirements analysts to figure out which rules have to be checked first, especially if the CRM is complex and there are numerous rules. In such cases we propose to manage inconsistency resolution incrementally by stepwise resolving groups of diagnostics rules. Freed from restriction such as checking of structural inconsistencies prior to static semantics ones, domain experts can drive inconsistency management in accordance with conceptual and methodological grounds rather than technical limitations.

5 Related Work

Zave and Jackson defined in [12] the scientific foundations of multi-formalism composition. They describe a formal CRM designed for formalizing a wide range of metamodels where composition is the conjunction of elements extracted from models. They describe a notion of functions dedicated to each input language for assigning input model semantics in the CRM. They then discuss a few methodological points on how to perform inconsistency checking. However, they do not discuss inconsistency types that can be verified and do not offer any implementation supporting their vision. More concrete composition solutions have been proposed as a way to perform semantic analysis [10,11]. In [10], the authors defines a framework to produce a HOL (High Order Logic) model from models expressed in different notations and analyze the composed model with HOL tools. Ainsworth et al. [11] propose a method for composing models written in Z. Relations between models are described with functional invariants and some Z proof obligations can be performed to verify composition correctness. However, these two approaches require that a significant amount of work shall be done manually, either for the pre-processing of Z models [11] or relevant information must be extracted and translated into a HOL specification by hand in [10]. Moreover, they do not process inconsistent models.

Kompose [14] is a meta-modeling approach built on the Kermeta language. It targets the automatic structural composition of aspects in software development processes. It is a rule-based approach where equivalence between objects is automatically calculated with regards to object structure. Kompose assumes that homogeneous input models (i.e. instances of a unique metamodel) are structurally consistent prior to their composition. Kolovos et al [20] propose a model merging language able to compose heterogeneous models which uses pattern-matching to merge models. However, similarly to compose they need to avoid conflicts before merge which restricts the inconsistency types that can be fixed and the global model and limit flexibility with respect to inconsistency management. Sabetzadeh and Easterbrook [9] provide an interesting composition framework based on a formal definition of models to compose. The composition operator (category-theoretic concept of colimit) is formally defined and traceability links

are automatically inferred. However this operator requires the model type of [21] which restricts highly the accepted IRLs. As opposed to Kompose and our approach, equivalence must be given by hand by the requirements analyst and composition only works for homogeneous models.

6 Conclusion

Dealing with inconsistencies across multiple models is a critical issue that requirements analysts and software engineers have to face nowadays. In this paper, we have shown how, by translating the problem from managing inconsistencies amongst heterogeneous models to managing inconsistencies within a single model, stakeholders' task can be greatly facilitated. In particular, we proposed a novel model composition mechanism which able to compose partial and possibly inconsistent models. Hence, various categories of inconsistencies can be checked on the resulting global model. Furthermore, as the order of inconsistencies to be solved is not prescribed by the approach, requirements analysts can flexibly drive the inconsistency management depending on the context. Native traceability supported by our implementation enables to report inconsistencies on the original models thus easing the determination of inconsistency causes. We are currently working on integrating our platform with formal analysis tools to obtain a complete requirements validation chain. Integration is performed by means of a model transformation translating CRM instances into models analyzable by the targeted tool.

In the future, we would like to acquire experience on the *adaptability* of the approach to various contexts and input languages. In particular, we will assess the impact of the introduction of a new input language on fusion rules and on the CRM.

References

1. Cheng, B.H.C., Atlee, J.M.: Research Directions in Requirements Engineering. In: FOSE at ICSE, Washington, DC, USA, pp. 285–303. IEEE Computer Society Press, Los Alamitos (2007)
2. Kotonya, G., Sommerville, I.: Requirements Engineering with Viewpoints. Software Engineering Journal (1996)
3. Rashid, A., Moreira, A., Araújo, J.: Modularisation and composition of aspectual requirements. In: AOSD 2003, Boston, Massachusetts, USA, pp. 11–20 (2003)
4. Easterbrook, S., Nuseibeh, B.: Using viewpoints for inconsistency management. Software Engineering Journal 11(1), 31–43 (1996)
5. Nuseibeh, B., Easterbrook, S., Russo, A.: Making inconsistency respectable in software development. Journal of Systems and Software 58(2), 171–180 (2001)
6. Nuseibeh, B., Kramer, J., Finkelstein, A.: A framework for expressing the relationships between multiple views in requirements specification. IEEE TSE 20(10), 760–773 (1994)
7. Nentwich, C., Emmerich, W., Finkelstein, A.: Flexible consistency checking. ACM TOSEM (2001)

8. Kolovos, D., Paige, R., Polack, F.: Detecting and Repairing Inconsistencies across Heterogeneous Models. In: ICST, pp. 356–364. IEEE Computer Society, Los Alamitos (2008)
9. Sabetzadeh, M., Easterbrook, S.: An algebraic framework for merging incomplete and inconsistent views. In: RE 2005, August -2 September, pp. 306–315. IEEE, Los Alamitos (2005)
10. Day, N., Joyce, J.: A framework for multi-notation requirements specification and analysis. In: 4th ICRE, pp. 39–48 (2000)
11. Ainsworth, M., Cruickshank, A., Groves, L., Wallis, P.: Viewpoint specification and Z. Information and Software Technology 36(1), 43–51 (1994)
12. Zave, P., Jackson, M.: Conjunction as Composition. ACM TOSEM 2(4), 379–411 (1993)
13. Brottier, E., Baudry, B., Traon, Y.L., Touzet, D., Nicolas, B.: Producing a Global Requirement Model from Multiple Requirement Specifications. In: EDOC, pp. 390–404 (2007)
14. France, R., Fleurey, F., Reddy, R., Baudry, B., Ghosh, S.: Providing Support for Model Composition in Metamodels. In: EDOC, Annapolis, MD, USA (2007)
15. van Lamsweerde, A., Letier, E., Ponsard, C.: Leaving Inconsistency. In: ICSE workshop on Living with Inconsistency (1997)
16. Baudry, B., Nebut, C., Traon, Y.L.: Model-driven engineering for requirements analysis. In: EDOC, pp. 459–466 (2007)
17. Nebut, C., Fleurey, F., Le Traon, Y., Jézéquel, J.M.: Automatic test generation: A use case driven approach. IEEE TSE (2006)
18. Spanoudakis, G., Finkelstein, A.: Overlaps among requirements specifications. In: ICSE workshop on Living with Inconsistency (1997)
19. van Lamsweerde, A., Darimont, R., Letier, E.: Managing Conflicts in Goal-Driven Requirements Engineering. IEEE TSE 24(11) (1998)
20. Kolovos, D.S., Paige, R.F., Polack, F.A.C.: Merging Models With the Epsilon Merging Language EML. In: Nierstrasz, O., Whittle, J., Harel, D., Reggio, G. (eds.) MoDELS 2006. LNCS, vol. 4199, pp. 215–229. Springer, Heidelberg (2006)
21. Corradini, A., Montanari, U., Rossi, F.: Graph processes. Fundamenta Informaticae 26(3-4), 241–265 (1996)

Experiences with a Requirements Object Model

Joy Beatty and James Hulgan

Seilevel, Inc.
3410 Far West Blvd., Suite 350
Austin, Texas 78731
{joy.beatty,james.hulgan}@seilevel.com

Abstract. [**Context and motivation**] Experiences in working with customers in the software development community have shown that the language used to talk about requirements is inconsistent. Because of this inconsistency, projects are struggling to develop products that meet the organizations' goals. [**Question/problem]** An inconsistent terminology leads to barriers to communication, which increases both the cost and length of time of development. In addition, the artifacts of requirements planning efforts are often ill-defined, and the team creates products that are not aligned with the organization's goals. [**Principal ideas/results**] As an attempt at resolving this inconsistent terminology and its fallout, this paper outlines the need for a common language. We propose a solution in the form of a Requirements Object Model (ROM) and study the use of the ROM in the requirements efforts on three software development projects. [**Contribution]** Evidence from these three projects demonstrates that the adoption of a common terminology leads to improved communication among project teams, and as a result, alignment about the business objectives for software development projects was achieved.

Keywords: Software Requirements, Requirements Models, Requirements Engineering.

1 Introduction

A recent study showed that over two-thirds of companies follow requirements practices which make project success improbable [1]. The CHAOS Report says that the top three reasons projects are failing are all related to requirements, specifically a lack of user input, incomplete requirements and changing requirements [2]. Reworking requirements during development or after deployment is extremely costly as compared to identifying the correct requirements during requirements phases [3]. The rework is expensive because it requires resources to handle the issues, results in lost revenue from the delay and causes delays to other projects**.** Furthermore, experiences with a variety of customers' software development projects have demonstrated that teams are often building software that is not what the business needs. Years of discussions with stakeholders have uncovered a common issue in that there is often no common understanding of project objectives among stakeholders. This leads to teams developing additional unnecessary features and sometimes even the wrong features.

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 104–117, 2009.

Further observations led to the realization that most projects do not have clearly articulated business objectives because teams do not know how to identify them and use them.

Working with customers has shown that the language with which requirements are talked about is inconsistent within an organization, which leads to barriers to communication. There are many different types of stakeholders that requirements practitioners deal with during the course of the project. For example, communicating a marketing manager's business needs for a feature to a developer can be challenging because there is no common language with which to label the various levels of requirements information. When the marketing manager communicates the business need, the requirements engineer might label it a business objective, while the developer thinks it is the justification of a single requirement. However, when the requirements engineer reviews the business objective with the marketing manager, it has a different name and is not recognized as an original need.

A common model, including both a language and approach to apply the language, is needed to help business users, requirements practitioners and development resources discuss and develop software products. A common model will enable teams to communicate better, as terminology will be consistent and understood by all stakeholders. Development resources will know the level at which they are receiving pertinent information. And all of the resources will understand how to use the common language to gather and use the information that drives the project priorities.

The ideas described in this paper are based on work initially done to help a customer create a standard requirements language to be used across their global organization. In addition, an approach was created for teams to use the language defined to help drive requirements for development. It is worth mentioning, a complete requirements methodology is important, however this paper is limited to the common language and approach to using this model to drive requirements and does not delve into details about how to execute the remainder of a methodology. The rest of the paper will further describe the need for a common model, explain and provide an example of a proposed Requirements Object Model (ROM) and discuss specific project experiences in which the ROM was used. The intent of this paper is to adapt what industry has already defined and alter it when necessary for additional clarity.

2 Review of Research on Requirements Terminology

Within and across teams, individuals typically have different understandings of the requirements terminology used. This is not surprising, given there really is no commonly agreed upon terminology for the various levels of the requirements hierarchy in the industry [4]. To illustrate this point, in one project example, a team member thought "need" referred to the justification for a feature; while another person thought "need" was the executive level justification for the project. When the team was asked to document "needs", it should be expected that there was a discrepancy in the type of information elicited.

There are various levels of requirements that must be explored and labeled with a common term. Some of the requirements information elicited will be very high-level and derive further detailed requirements, while other information elicited will be

detailed requirements for development. For example, a business user may describe a need to achieve a revenue target for the year, have a replacement call-center system, be able to view the upcoming call queue and have specific colors of buttons on the screen. These are pieces of information at very different levels of the requirements hierarchy. All of them are very important to understand to achieve project success; and therefore, a common set of names would enable easier communication about them on the team.

Experts in industry have defined terms of their preference. The following describe some of these definitions to help illustrate the variances.

- "Business requirements" or "business objectives" are used to describe why a product is being built, "user requirements" describe what the user wants and "functional requirements" describe expected system behaviors the developer should build. "Features" are logical groupings of functional requirements [5].
- "User requirements" are user-imposed constraints or affordances, and "functional requirements" are things the system is required to do. This author also makes use of the term "goal" to describe desired results, often decomposed through various levels until they are written as use case titles [6].
- "Business objectives" are goals of the business or customer requirements. "Business requirements" are different, in that they are essential activities of the organization and are derived from the business objectives [7].
- "Business requirements" are an answer to business problems, "user requirements" solve user specific problems, and "software requirements" describe the software solution. In this case, user requirements are a bridge between the business requirements and software requirements [8].
- "Business requirements" can be used interchangeably with enterprise goals, objectives, or needs. "User requirements" describe stakeholder needs. "Functional requirements" describe the system behavior and information it will manage [9].
- "Need" is the highest level of information, with "goals" and "objectives" being derived from the needs. "Goals" describe what the product will do, while an "objective" further describes how the goals will be met [10].

It is apparent that these terms are very closely related, but that the literature does not adopt a common terminology with precisely the same definitions. Business requirements, business objectives, goals and needs mean the same thing in some cases, but not always. Functional requirements and software requirements mean the same thing in some cases, but vary in other definitions.

3 The Need for a Common Model

Experiences show that most teams do not start projects with clear goals, but rather make assumptions about the business needs and dive into requirements writing. Typically, the business goals are defined higher in the organization and passed down to the project team, sometimes implicitly. In fact, in training requirements engineers, it has become clear that most of them do not know what their projects' business objectives

are or how to identify them. Furthermore, teams often make up their own definitions about what the requirements terminology means.

Initial evidence shows that a common set of terminology might allow teams to better collaborate on requirements. The team members would not have to spend time thinking about what their co-workers meant by phrases such as "business requirements" or "needs". If working with a common language, when developers receive requirements information, they will know at what level they are receiving it – whether a high-level introduction of scope or low-level requirements to develop. This paper suggests that adoption of a model, which includes a common terminology and an approach to apply it, will help improve the overall requirements methodology. Through the adoption of a common model, it will be easier for project teams to identify missing pieces of information necessary to the development lifecycle, as well as features that have not been given proper business justification.

4 ROM Defined

4.1 Overview

The ROM is a suggested model that defines language for talking about requirements and an approach to apply the language to projects in order to build successful products. It provides a common language with which to communicate globally about the requirements and an approach to encourage early identification of an organization's goals.

The ROM has multiple levels of objects. In order to further discuss these objects and how they relate, it is first necessary to define them. To clarify the definitions, other common phrases with the same definition are included.

One intention in designing the model was to make it easy to learn and adapt. Therefore, in some cases, the terminology used in industry was explicitly avoided in the proposed model in order to avoid confusion with these terms.

4.2 Business Level Definitions

Business level objects are entities in the requirements activities that are defined early in projects and relate to the organizational priorities. They are not specific to any product or project, but to the business as a whole.

Business Problems. Describe problems that are preventing the business from achieving its goals.

Business Objectives. Desired metrics the business seeks to meet in order to solve the problem. Business objectives may also be commonly referred to as sales targets, senior management's goals, business requirements, or goals.

Business Strategy. An approach to meet the business objectives. A business strategy is not specific to any one product or project solution. In most projects dealing with requirements engineering, an adopted strategy typically involves building a system or software.

4.3 Project Level Definitions

Project level objects should only be defined once the business level objects have been established and a business strategy leads to a specific project. These are the objects which are derived from the business level objects and which lead to the development of a software product.

Product Concept. Proposed solution derived from the business strategy to accomplish one or more business objectives. This becomes a project and can be described with a project vision or mission statement.
Success Metrics. Statements about the specific desired measurable outcomes to meet the business objectives. If there is only one project to meet the business objectives, then the success metrics are likely the same as the business objectives. These may also be called project management targets or product targets.
Guiding Principles. The approach to meet the success metrics for the project. They are common themes to be considered in creating a product that will drive the feature set and specific requirements. They may be statements of market desires, stakeholder goals, user requirements, or product strategies.
Product Features. A collection of functionality that provides a set of services to the users. System requirements are gathered for product features.
Product Qualities. A collection of desired qualities about the product. Product qualities encompass non-functional requirements.

4.4 Relationships between the Levels

An object model is more than just a language; it also defines how the objects in the model relate to one another. As such, the ROM is divided into a hierarchy (see Fig. 1) where objects lower in the hierarchy can be said to have a "derived from" relationship to objects higher in the hierarchy. For example, features and product qualities are

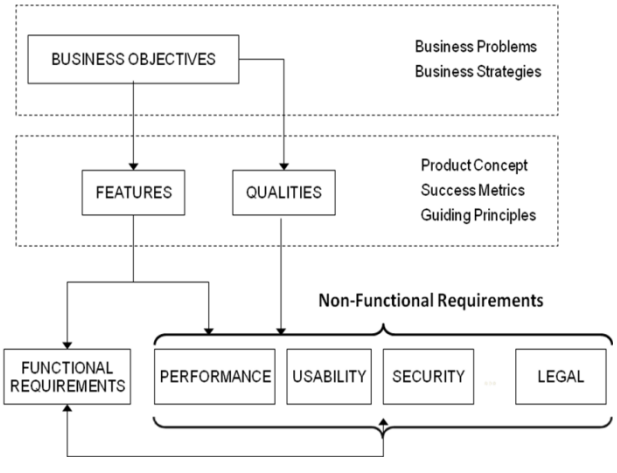


Fig. 1. Relationships between objects in the ROM

derived from business objectives. The product concept, success metrics and guiding principles should be defined to help derive features. Functional requirements and the other requirement types all derive from the features and qualities.

4.5 The Hierarchy Repeats

There are often many levels of the objects in the ROM repeated (see Fig. 2). For example, at the beginning phases of a new project during a discussion with a director, the director may describe a business problem, with business objectives and business strategies to solve this problem. Over the course of this discussion, another business problem is discovered, which also has business objectives and business strategies employed to solve this problem. It is important to define these layers starting at the top and going down enough layers—low enough to get to a project level. Similarly, the project level may also contain nested ROM hierarchies—each project may define its own business level objects from which to derive project level objects for the product being developed. If this is the case, then it is important to ensure that the business objects defined at the project level are consistent with the higher-level business objects.

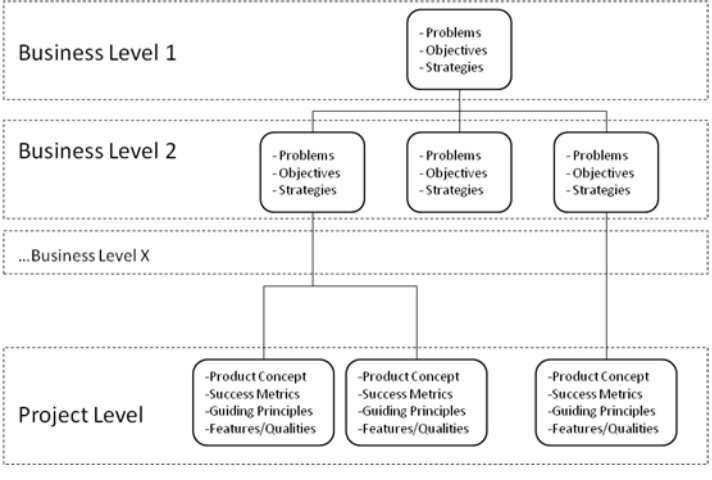


Fig. 2. The ROM Hierarchy repeated over several different levels

5 Using the ROM

5.1 Typical Approach

Experiences have demonstrated that most projects start with the equivalent of a product concept. The project team might define success metrics, such as a launch date. However, they quickly jump into defining features and qualities and then proceed with requirements and design. The problem with the typical approach (see Fig. 3) is that there are no agreed upon business objectives with which to guide the project scope, and development strays from the intended goals of the business.

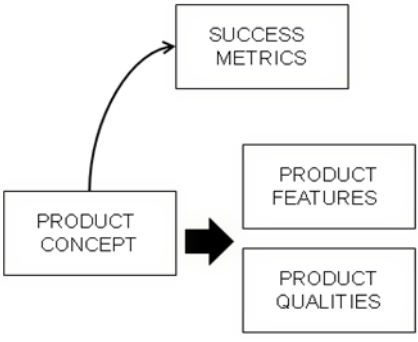


Fig. 3. Typical Approach

5.2 Ideal Approach

An ideal approach (see Fig. 4) would be to start with the business level problem and define objectives and strategies which solve that problem. The project level would then be defined, leading to the product requirements. This approach is ideal because it ensures that the features developed are always driven by the solution to a business problem. The issue with this is approach is that it does not reflect the reality of how organizations operate, in that most organizations are more chaotic and less linear in thinking about projects. However, the main reason this approach does not work is that requirements engineers are typically engaged well after the business level thought processes have evolved.

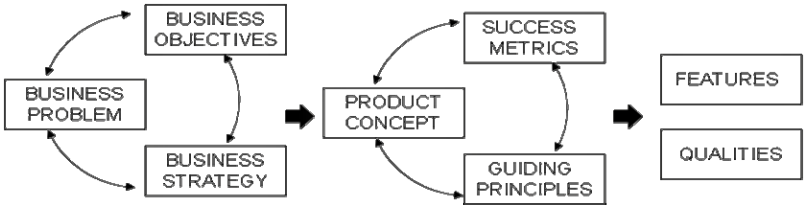


Fig. 4. Ideal Approach

5.3 A Realistic Approach

Many projects will continue to start with a product concept, simply due to the timing of when the requirements team is engaged. Therefore a top-down approach to start at the business problem is not realistic; however the same levels can be applied to achieve success (see Fig. 5).

Most projects start with a product concept (e.g., "Replace the call center application"). Frequently, a set of features is also already defined (e.g., "Viewable call queue"). However, in this case, working up from the product concept, the problem that is being solved can still be understood. By asking "why is that a problem?", eventually a problem will emerge that relates to money. It should be noted that "relates to money" in the context of business often means relating to revenues, profit and costs to

the company. Each project is unique, so this is just a guideline to identify the business problem. In fact, in some cases the business objectives maybe be "improved patron service", however most organizations will have an objective above this that relates to money, such as "retain 30% of new customers" which is derived from "increase revenue by 5%". Depending on the level of the organization in which the practitioner is working, the project's relationship to revenue, profit or cost savings may not be initially clear. However, the practitioner should elicit this information from the appropriate personnel, often needing to work higher in the organizational structure, in order to clearly define the project business objectives. From a clearly defined problem and desired business objectives, appropriate strategies can be derived that meet the business objectives.

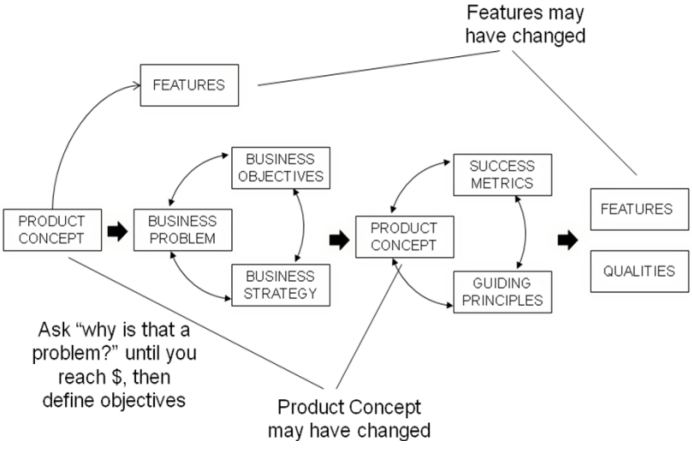


Fig. 5. Realistic Approach

Using the business problem, objectives and strategies, a clear product concept can be developed, which may or may not be the same as the original product concept. Success metrics and guiding principles should be developed for the project, followed by features and qualities. Similarly, the features may have changed from the original suggestions, in order to map back to actual business objectives.

Establishing business objectives is critical for the success of a project, because they are used to define and control the project's scope. Product concept, success metrics and guiding principles represent steps that help to make the bridge from business objectives to features.

6 Questions to Help Complete the ROM

As described previously, there are different levels of requirements. However, when eliciting requirements, users do not distinguish between these different levels, nor should they. The requirements engineer has to take in all of the levels of information provided and organize them. Additionally, they must recognize when various levels of

the ROM are not provided and explicitly elicit those. There are questions (see Table 1) that requirements engineers can ask of various stakeholders in order to complete the ROM and identify each of the objects.

Table 1. Questions to ask to complete the ROM

ROM Term	Questions to Ask
Initial Product Concept	What are we building?
Business Problem	What problem does this product solve? Continue asking the question "why is that a problem?" until money is in the answer.
Business Objectives	What metrics can we use to determine that we have solved the business problem?
Business Strategy	What are the approaches that could be followed to meet the business objectives, thereby solving the business problem? What problems or aspects of problems are being solved and in which way? What stops you from solving the problem today, and what is your strategy to resolve that?
Product Concept	What products must be built for each of the business strategies, in order to meet the business objectives?
Success Metrics	What metrics can we use to determine if the product built is a success? How will we know if it fully enables its intended contribution to the business objectives?
Guiding Principles	What are the overarching themes for the product that should be considered in developing the product? What high-level market desires should influence the product? What is the overall approach/strategy in designing the product to ensure the success metrics are met?
Product Features	What buckets of functionality are needed based on the guiding principles and users' requirements? What features are driven by the high-level models (i.e., process flows, use cases,)?
Product Qualities	What qualities about the product are important to develop given the guiding principles of the project? If the functionality is built correctly, what types of things would keep users from adopting the software?

7 ROM Example

An example of the ROM in a mock-project will help further explain it. In the example, a team was assigned the project of building an online Yahtzee® game. Table 2 demonstrates a set of questions and answers used to identify a business objective and business strategies.

The series of questions led to a clear business problem, followed by a business objective. Taking the questions further, business strategies were identified. There are often many business strategies that can be defined to achieve the business objectives. However, a project stakeholder will need to make a selection about which ones will be implemented. In this example, it turns out that building an online Yahtzee® game is a valid product concept for the first strategy, as long as it is a product designed for seven to thirteen year olds.

Table 2. Identifying Example Business problem, business objective and business strategies

Question	Answer
What is the Product Concept?	An online Yahtzee® game
Why would you build Yahtzee®?	Our current games are complicated
Why do you care if your games are complicated?	We are not selling outside the age group of 15-30 year olds.
Why do you care if the rest of the market buys?	Sales to the 15-30 year old market are slowing down.
Why are you concerned about that growth?	Our competition is still growing, while we are not. (Business problem about "money" identified)
What kind of growth do you want?	25% growth in markets outside of 15-30 year olds. (Business objective identified)
How can we target growth in other markets?	Build a game for 7-13 year olds. Advertise to retirement age people. (Two business strategies identified)

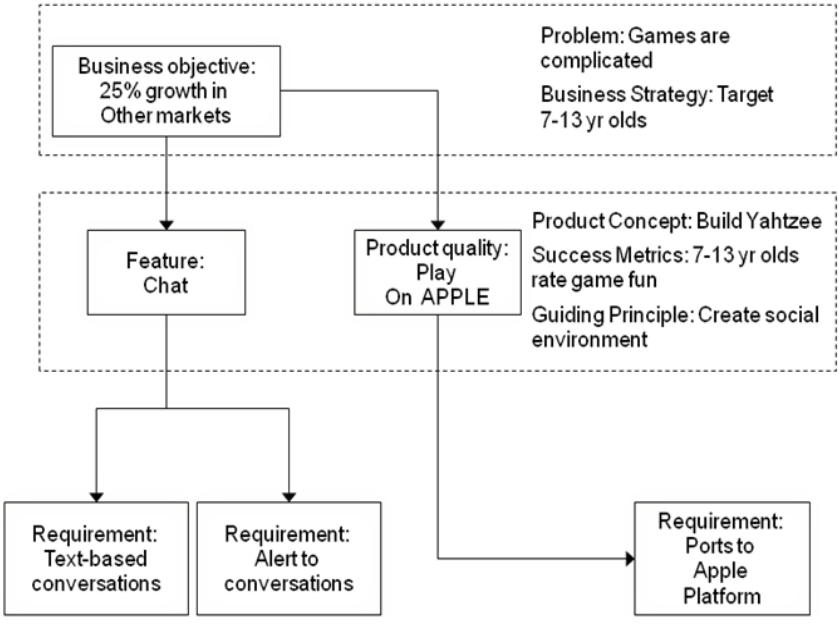


Fig. 6. Example ROM

To further explain this example, using the business objective, business strategy and an understanding of the detailed problem, a product concept can be more clearly defined, along with success metrics and guiding principles for the project (see Fig. 6). For example, the product concept was "Online Yahtzee®", a success metric was "7-13 year olds rate the game as fun on a standard survey", and a guiding principle was "create a social environment". From these, features and qualities were defined,

ultimately to derive requirements. An example, a feature was “chat sessions” and a product quality was “they must be able to play it on an Apple”. These can be further broken into requirements such as “System shall support text-based conversations” and “System shall alert user to any incoming conversations”.

8 Project Experiences

The following examples describe experiences with using and not using the ROM on requirements engineering projects. Perhaps the most important lesson learned from these experiences is that quantifiable success metrics are critical for project success. Additionally, a weakness of this approach is that factors unrelated to the development of an IT system can contribute to project success or failure.

8.1 Pricing Analysis Example

In this project, development had been underway to replace the backend of a pricing analysis tool. The project had gone in several directions with very little focus, and the schedule had slipped a month due to this lack of project focus. When the requirements team became engaged with the project, they began defining objects within the ROM for this project. Requirements practitioners first spoke with the development manager to determine which business problems and objectives were driving the development of the pricing tool. The initial statement of the business objective was to “build a global pricing tool”, which exemplifies how the project was not focusing on solving a particular problem related to money. Through the course of these discussions and attempting to understand the business problem, the team uncovered other business problems that management was attempting to solve using this tool. Helping define the business level objects for the project helped stakeholders prioritize features, as stakeholders were able to quickly see which features solved business problems and met business objectives. For example, some of the business problems identified were that the system needed to be in compliance with certain Securities and Exchange Commission (SEC) controls. The lack of this compliance had led to decreased shareholder confidence and subsequently, decreased investment revenue. The definition of these problems helped the business determine which problem was most affecting their business, which in turn ultimately drove the features of the pricing tool which best solved their most important business problem. Often, during the discovery of business problems, requirements and non-functional requirements for the system were discovered in the discussions immediately. For example, one of the reasons no one used the current system was that the server response times for pushing updates were horrendously long.

A benefit of the ROM, according to the project’s requirements expert, is that during elicitation and prioritization sessions, the requirements team can ask everyone in the room if they are doing the right things, developing the right features and help guide people towards what the objective is. Since the ROM can ideally fit on one page, it can also help get all members of the team—from business users to development resources—to understand the problems being solved.

Some challenges encountered were that the practitioners had difficulties convincing the IT manager of this approach, even though they were able to successfully

convince the business manager of the benefit of using the ROM and insist on using success metrics and business objectives to drive feature prioritization. Indeed one of the biggest challenges is getting buy-in from stakeholders with multiple interests to use the ROM approach. This is further complicated when the requirements team is engaged on a very large project that is in mid-development with previous lack of focus and priorities.

8.2 Loan Originations Example

This project was to evaluate current and desired functionality of loan origination software, so that the system could be replaced by a new system, because the existing system could not be further updated. When the practitioner joined this project, a ROM had been partially completed; however, success metrics had not been defined. The development team had jumped directly to the product concept without defining metrics by which to measure the success of the product. Indeed, success was defined circularly by several stakeholders ("We will be successful as soon as we build the product"). Asking the question, "What does success look like?"—and tying it to the solution of a business problem—allowed the project to maintain focus. During stand-ups and other meetings where features were prioritized, practitioners and managers were able to ask themselves, "Is my team doing all and only those things which support project success?" This is critical, since without having a good idea of what counts as success, one cannot do things in the direction of success. Additionally, since projects do not have unlimited resources, defining project success and business objectives helps ensure that limited resources remain focused on the most critical business needs. The requirements practitioner on the project suggested that software development can be seen as a process of change. The most difficult thing about change is helping the teams maintain focus—and to keep people focused on the important things. Success metrics are about maintaining focus on the project. A successful outcome of this project was that the business manager was able to have better conversations with her superiors, allowing her to win support for the project by giving her a language with which to justify the project to both her managers and the teams she supervised.

A challenge of this approach as stated by the practitioner, is that even though the various teams were able to focus on the problems being solved and how to solve them, there are still things outside of the control of the practitioner (for example, hiring decisions for development resources). In this way, applying the ROM did not always guarantee that the project would be successful.

8.3 Semiconductor Factory Example

This project evolved over the course of several years and involved several different development teams dispersed geographically and culturally. The goal of the project was to develop a system which would eventually automate a semiconductor factory.

When the project started, there was one mission, which was to "select a tool to run the fab in our custom way". There were business objectives above this, such as increasing yield or decreasing the cost per wafer, but at that point in the project, those were unstated to the project team. As a result, the team immediately jumped into eliciting

desired features of the tool, but did not pay attention to time to deploy the tool or what features would help meet the business objectives, since these were unclear.

A couple of years into the project, practitioners were able to present the value of using business objectives to drive feature prioritization at the beginning of a release cycle to IT managers and the system architect. For the first time in the project history, managers defined business objectives and success metrics for each release and used these to drive scope discussions and prioritization. Very quickly, management began thinking in terms of the ROM and was able to use the objects in the ROM to drive feature development, as well as to justify development budgets to their management. Naturally, during this transitional period, the challenges were to acclimate stakeholders to the new terminology and concepts being discussed.

With each release that followed the transition period, the architect and stakeholders became used to thinking in terms of the ROM. The development process was modified to accommodate the ROM, with success being clearly defined and quantifiably measured following each release. Consequently, the ROM helped frame scope discussions in terms of the business objectives to be achieved. Practitioners noticed that by using the ROM, prioritization and scope definition became easier with each release. An added benefit was that requirements traceability was more easily achieved, since the ROM helped define the traceability relationships and the objects to be traced.

One of the challenges on this particular project was that stakeholders (in this case, users), after learning what the business objectives and success metrics were, would sometimes use these as an ad hoc justification for features that they had been requesting for, in some cases, years. This led to some artificial justifications for certain features. These ad hoc justifications were not a result of dishonesty or manipulation on the part of the stakeholders, but of a real belief on their part that a particular feature would help achieve a business objective. In this case, the value of the success metrics is clear, as they help determine whether the feature did, in fact, achieve the business value (even if it is after the fact). Additionally, when practitioners were engaged with a project for this length of time, other business strategies not related to the development of an IT system were revealed. For example, a way to increase revenue might be to attempt to license technology or to acquire one's competition through a merger. In these cases, it is difficult for the ROM to accommodate discussions around business strategies which do not involve building new software. These strategies ideally would have been discovered prior to committing to a particular software project.

9 Conclusions and Future Work

The ROM is a proposed requirements hierarchy and approach to apply it to help project teams deliver more successful projects. Initial experiences demonstrate that a common model will enable teams to communicate better with a common language and focus teams on developing products that are within scope of the business' goals.

The ROM was initially developed out of discussions with a customer team, many practitioners and two industry experts to address the customer's global terminology gaps and lack of formal approach to defining the initial high-level requirements. It has been used on multiple projects since; however it is necessary to continue to test the ROM on additional projects. As a result of further testing, the terms in the ROM may

need to be updated if they prove to be unclear. Additional levels or terminology may be added to the ROM if any are determined to be insufficient. Also, the approach to apply the ROM to projects may need refinement and a method of training teams to use it. Tools to facilitate capturing the ROM on projects might be beneficial and represent an area for expanding the research, including understanding what tools already exist to aid using ROM and where there are weaknesses in those tools that must be improved. Although not investigated thus far, using the ROM may have a nice impact in the verification and validation of software solutions. Finally and most importantly, any discussions and feedback on the proposed ROM from industry experts would be greatly appreciated, leading to further refinements of the model.

Acknowledgement

The authors of this paper would like to express their gratitude to Ian Alexander and Karl Wiegers for their valuable discussions and feedback on the proposed terminology and approach in their early stages.

References

1. IAG Consulting: Business Analysis Benchmark Report (2008), http://www.iag.biz
2. The Standish Group: The Chaos Report (1994), http://www.standishgroup.com
3. Boehm, B.: Software Engineering Economics. Prentice-Hall, Englewood Cliffs (1981)
4. Alexander, I., Beus-Dukic, L.: Discovering Requirements: How to Specify Products and Services. Wiley, Chichester (2009)
5. Wiegers, K.: More About Software Requirements. Microsoft Press, Redmond (2006)
6. Alexander, I., Stevens, R.: Writing Better Requirements. Addison-Wesley Professional, Redmond (2002)
7. Young, R.: Effective Requirements Practices. Addison-Wesley, Boston (2001)
8. Gottesdiener, E.: Requirements by Collaboration. Addison-Wesley, Boston (2002)
9. International Institute of Business Analysis: Business Analysis Body of Knowledge (2008), http://www.theiiba.org/AM/Template.cfm?Section=Body_of_Knowledge
10. Hooks, I., Farry, K.: Customer-Centered Products. AMACOM, New York (2000)

Architecting and Coordinating Thousands of Requirements – An Industrial Case Study

Krzysztof Wnuk[2], Björn Regnell[1,2], and Claes Schrewelius[1]

[1] Sony Ericsson, Lund, Sweden
http://www.sonyericsson.com
[2] Lund University, Sweden
{bjorn.regnell,krzysztof.wnuk}@cs.lth.se
http://www.cs.lth.se

Abstract. [**Context & motivation**] When large organizations develop systems for large markets, the size and complexity of the work artefacts of requirements engineering impose critical challenges. **[Problem]** This paper presents an industrial case study with the goal to increase our understanding of large-scale requirements engineering practice. We focus on a senior requirements engineering role at our case company, called requirements architect, responsible for quality and coordination of large requirements repositories. **[Results]** Based on interviews with 7 requirements architects, we present their tasks and views on architecture quality. **[Contribution]** Our results imply further research opportunities in large-scale requirements engineering.

Keywords: Large-scale requirements engineering; Empirical study; Requirements repositories; Requirements dependencies; Requirements architect.

1 Introduction

Large software companies are often confronted with large and complex requirements repositories. The requirements originate from multiple sources and address multiple customers and market segments. This paper presents an industrial case study of the tasks involved in managing large and complex requirements repositories. The investigated tasks are related to a role called *Requirements Architect* that has recently been introduced at the case company. The requirements architects are responsible for the scope of large platform projects that products are based on [4]. Our motivation to perform this study was to understand current practices in working with large-scale requirements repositories and to find issues for future research. In this study, we have conducted interviews with requirements architects in order to address the following questions: (1) What are the tasks related to working with large-scale complex requirements repositories on multiple products platform projects? (2) How do practitioners perceive the notion of *requirements architecture* and how do they describe good requirements architectures?

The second question is related to sustainable requirements architectures [5]. With the term requirements architecture we mean the underlying structure of requirements, including the data model of requirements with their pre-conceived and emerging

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 118–123, 2009.

attributes and relations. By sustainable architectures we mean structures that allow for controlled growth while allowing requirements engineers to keep track of the myriad of issues that continuously emerge. Practitioners facing a transformation to large-scale requirements engineering (RE) may use this research to gain insights in what may come, and researchers may use the results to inform their choices of future research directions.

The paper is organized as follows: Section 2 describes the industrial context at the case company. Section 3 provides the methodology description. Section 4 and 5 highlights the result of interviews. Section 6 concludes the paper.

2 Industrial Case Context

The interview study was performed at Sony Ericsson. Due to the technological complexity of the domain, the case company is working in parallel in many advanced system engineering areas such as radio technology, audio and video, and positioning. The complexity of requirements engineering is driven by a large and diverse set of stakeholders, both external to the company and internal. Different stakeholders have different demands on the future functionality of the mobile phone which they express by different types of requirements. Requirements originating from external stakeholders are called *market requirements.* They are mainly supplied by mobile operators, which usually submit specifications with thousands of requirements that require gap analysis. Other sources of requirements are the *Application Planning* and *Product Planning* departments. The platform and market requirements also have to be checked against supplier requirements to ensure that certain functionality can be delivered by a corresponding platform project, including integration of subcontracted parts. Currently, the case company's requirements database contains around 30 000 platform system requirements and a few thousands supplier requirements. The platform system requirements are organized into *features* that represent the smallest units that can be scoped into or out from the platform project [2]. The case company develops products using a product line engineering approach, where one platform project is the basis for many products that reuse the platform project's functionality and qualities [4]. Within the platform project, the case company has defined a number of requirements engineer groups called *Technical Working Groups* (TWGs). They are responsible for elicitation, specification and prioritisation of high-level requirements within a specific sub-domain. Within this industrial context, requirements architects work mainly with platform system requirements and features. Their main responsibility is the management of the scope of platform projects by helping TWGs to specify requirements and project management to see all implications of the scoping decisions. The scoping decisions are made by a *Change Control Board (CCB).*

3 Research Methodology

To study individual perceptions of requirements architect role at the case company, we conducted seven semi-structured interviews [6]. Before conducting interviews, a brainstorming and planning meeting was conducted. During this meeting, the scope of

the study was agreed upon and an interview instrument was developed with a set of questions, where the wording could be changed and the order could be modified based upon the interviewer's perception [6]. The third author, acting in his role as manager for requirements architects at the case company, participated in the development of the interview instrument and invited seven interviewees with various experience within the requirements architect role. These persons were chosen from three sub-organizations within the case company, each responsible for products for different market segments. It was sent out via email to all the participants in advance and also discussed at the beginning of each interview to ensure that the scope of the interview was understandable. The interviews were held during the autumn of 2007 and varied in length between 60 and 110 minutes. All interviews were attended by two interviewers and one interviewee. Questions were kept simple and effort was put on avoiding leading or biased questions [6]. All interviews were transcribed. After transcription, each of the interviewees received the transcripts for validation. Interviewees analysed their transcripts in order to ensure that the interviewers heard and understood the recordings and notes correctly. In case of misinterpretations, corrections and comments were sent back to the researchers. The data was then imported to a spreadsheet program to perform a content analysis [3] based on categorisation. The categories such as tasks or notion of requirements architecture quality, were chosen based on the interview instrument topics and other emerging topics in the interviews. Additionally, for each category notes describing problems and improvements were added. Finally, the results were validated by two interviewees that gave independent comments to the proposal of the tasks derived from the interviews.

4 Tasks of the Requirements Architect in the Case Company

Based on the analysis of interviews, we have identified six tasks, listed in Table 1, that represent what is considered to be important obligations of the requirements architect role when acting as a senior coordinator in a large-scale setting. Several tasks (T1, T4, and T5) are directly related to change management. In order to cope with the initial definition of the platform projects scope and later incoming change proposals to the platform projects, requirements architects facilitate communication across

Table 1. Tasks and goals for requirements architect in the case company

Task	Goal
T1: Scope management	Ensure that the platform project scope changes are addressed and that the change proposals are prepared.
T2: Gap analysis	Ensure that misalignments between market requirements and supplier requirements are addressed.
T3: Enforce requirements quality improvements	Check the quality of requirements. Alert if requirements quality improvements are needed.
T4: Drive CCB investigations	Drive change proposal investigations in order to gain understanding of the impact of the scope changes.
T5: Present the scope	Present the scope of the platform project at milestones.
T6: Request requirements architectures improvements	Ensure that the requirements structure is maintained according to defined rules.

different groups of requirements engineers. This may indicate that the complexity in both requirements inter-dependencies and organisational structure in the large-scale case imply hard challenges in communicating decisions about changes. The analysis of gaps between market requirements and what is offered by technology suppliers (T2) is increasingly complicated as the number of stakeholders on the market increases and the number of technical areas that are covered gets larger.

Also, for a basic and common task such as checking the quality of requirements (T3), interviewees express challenges related to the cohesion of complex multilayered requirements structures that originate from multiple sources. In our case, requirements architects have to drive complex changes (T4) that span over many technical areas and may impact many product releases in one platform. Another challenge related to these investigations is the ability to ensure that investigations are made by the right persons with the right competence and that the full impact picture will be ready before CCB decision meetings. Missing some of the aspect may have a great impact on the whole platform project. In a large scale case, the task of presenting the current scope (T5) is especially demanding as the requirements architect must understand both technical aspects as well as the business and market impact of all features in order to conclude them in a way that is meaningful to high-level management and marketing. Finally, we report that in a case like the one we have examined, where several parallel large platform projects coexist, there is an expressed need for a person with a holistic view that has a mandate to request requirements architecture improvements (T6). In this case, the responsibility for ensuring architectural consistency of requirements is not delegated to the projects, but is managed across projects by requirements architects.

5 Views on Requirements Architecture and Its Quality

In our interviews with practitioners we have confirm our pre-understanding that the concept of requirements architecture is complex and include many aspects. We have deliberately not imposed a pre-conceived, closed definition of the concept on our interviewees, as we wanted to base our understanding of the requirements architecture on empirical data. We cannot say that a single, generally accepted definition of requirements architecture has emerged, but our findings indicate that all interviewed practitioners included some of the following aspects in their views on requirements architecture: (1) the requirements entities themselves (such as features, system requirements, detailed requirements, functional requirements, quality requirements, etc) and their relationships; (2) the information structure (meta-model) of requirements entities including (a) attribute types of entities, and (b) the relationship types including different types of dependencies to other entities; (3) the evolution of the information structure (a) over time and (b) across abstraction levels as entities are refined both bottom-up and top-down; (4) the implications of organisational structures on requirements structures; (5) the implications of process and methodology on requirements structures; (6) the implementation of tool support and its relation to requirements structures, organisation, process, methodology etc.; (7) the scalability of the requirements structures as the number of entities increase and the interrelated set of entities gets more complex.

In our interviews with requirements architects, we also discussed the notion of quality of requirements architectures. We started the discussion based on the analogy of how system architecture quality supports good design and implementation of systems, and transferred this analogy to how requirements architecture quality supports good requirements engineering. The following quality issues were identified when analysing interview transcripts:

Understandability and cohesion. Responders expressed the opinion that a good requirements architecture should be easy to understand and designed to enable a holistic view of different types of modules and abstraction levels in order to enable easy identification of vital information. Furthermore, the way how the structure of requirements information is visualized was also mentioned by our responders as an important factor influencing mentioned quality issues.

Robustness, integrity and enforcement of policies. An established process for managing and architecting requirements can result in a consistent, reliable and robust requirements architecture. Lack of clear policies and working rules may result in low reliability of requirements as well as discrepancies in usage of the architectural policies across projects.

Extensibility, flexibility and efficient traceability. According to our responders, a good requirements architecture should allow for controlled growth by being extensible and flexible without endangering the previously mentioned qualities of robustness and integrity. Cost-efficient traceability among requirements at different levels of abstraction when continuous growth and refinement occur is important. A good balance between extensibility, flexibility and traceability on one hand and the complexity driven by these qualities on the other hand has to be achieved in order to avoid the risk of ending up with an unmanageable repository.

6 Conclusions

This paper presents tasks related to a role called requirements architect, which is working with large and complex requirements repositories at the case company. We also present practitioners views on quality attributes of the artefact called requirements architecture. Efficient management of large sets of information is considered to be crucial in many disciplines. Similar to software architecture, the information model is considered to be not only a technical blueprint for a software-intensive system, but it also includes social, organisational, managerial and business aspects of the software architecture [1]. At our case company, the requirements architecture is an artefact that is managed separately, but in relation to the system architecture, and interviewees express a range of issues that need to be addressed, both soft issues such as organisation and business models as well as technical aspects.

The requirements architect role at our case company is motivated by a perceived need of special attention to cross-cutting issues, and inter-disciplinary communication across sub-domains and technical areas. We found several tasks of normal requirements engineering practice, such as change management, scoping and specification quality enforcement that is viewed as particularly challenging in the studied large-scale setting, and therefore included in the responsibilities of requirements architects acting as senior coordinators of the requirements engineering process. We also found

expressions for specific quality aspects of the requirements architecture itself that are viewed as important to support an effective and efficient management of an increasingly large and complex repository.

In relation to the concept of requirements architecture, we highlight the following areas to be considered in further research:

- Continued conceptual and empirical investigation of the notion of requirements architecture.
- Investigations on features of computer-aided tools for managing requirements architectures.
- Studies of the organisational and process aspects in relation to requirements architectures.
- Development of assessment instruments for requirements architecture quality and competence certification of requirements architects.
- Analysis methodology and visualisation models for requirements architectures in large-scale product line engineering.

Acknowledgements. This work is supported by VINNOVA (Swedish Agency for Innovation Systems) within the UPITER project. Special thanks to the anonymous interviewees for their valuable time and knowledge. Thanks also to Thomas Olsson and Lena Karlsson for the initial input on a draft version of this paper, and to Lars Nilsson for valuable language comments

References

1. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice, 2nd edn. Addison-Wesley, Reading (2003)
2. Clements, P.: Being proactive pays off. IEEE Software 19, 28–30 (2002)
3. Patton Quinn, M.: Qualitative Research & Evaluation Methods, 3rd edn. Sage Publication Ltd., London (2002)
4. Pohl, K., Bockle, G., van der Linden, F.J.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer, Heidelberg (2005)
5. Regnell, B., Berntsson Svensson, R., Wnuk, K.: Can We Beat the Complexity of Very Large-Scale Requirements Engineering? In: Paech, B., Rolland, C. (eds.) REFSQ 2008. LNCS, vol. 5025, pp. 123–128. Springer, Heidelberg (2008)
6. Robson, C.: Real World Research, 2nd edn. Blackwell Publishing, Oxford (2002)

BPMN-Based Specification of Task Descriptions: Approach and Lessons Learnt*

Jose Luis de la Vara and Juan Sánchez

Centro de Investigación en Métodos de Producción de Software,
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022, Valencia, Spain
{jdelavara,jsanchez}@pros.upv.es

Abstract. [Context & motivation] The need of organizational modelling during the requirements engineering process of an information system has been widely acknowledged, and business process modelling can be considered a must. Nonetheless, the specification of functional requirements can be inadequate if business processes are not properly analysed so as to elicit these requirements. **[Question/problem]** There is a gap between business processes and functional requirements that must be bridged in order to specify the functional requirements of an information system. In addition, means of precisely and homogeneously elicit these requirements from business processes are necessary. **[Principal ideas/results]** The goals of this paper are: 1) to present an approach that provides methodological guidance to properly specify functional requirements from business processes; and 2) to report on practical experience using the approach. The approach is based on the analysis and graphical enrichment of BPMN diagrams for the elicitation and specification of functional requirements in the form of task descriptions, and it has been applied in field trials with a software development company. **[Contribution]** The main contributions of the paper are: 1) the extension of BPMN for proper elicitation of task descriptions; 2) the provision of detailed guidance in order to adequately use BPMN diagrams for the specification of task descriptions; and 3) the presentation of the lessons learnt by using the approach.

Keywords: Information System, Requirements Elicitation, Requirements Specification, BPMN, Task Description.

1 Introduction

Understanding of the application domain is essential for the requirements engineering (RE) process of an information system (IS) for an organization. Organizational concerns must drive requirements elicitation [24], and requirements must be defined in terms of phenomena that occur in the organizational environment [28]. As a result, the need of organizational modelling during the RE process has been widely acknowledged for the last two decades (e.g. [3][9][12][22]).

* Research supported by the Spanish Government under the project SESAMO TIN2007-62894 and the program FPU AP2006-02324, and co-financed by FEDER.

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 124–138, 2009.

Business process modelling is part of most of the organizational modelling-based RE approaches, and it can be considered a must for IS development. ISs for organizations should manage and execute operational processes involving people, applications, and/or information sources on the basis of process models [11], and RE approaches for their development should differ from traditional ones [1]. First, detailed process models are necessary in the RE process. Second, new systems must support new ways of (better) running an organization.

Once the desired business processes of an organization have been modelled, they must be analysed in or order to bridge the gap that exists between business and (software) system domains. This gap is the consequence of characteristics such as different terminology, levels of granularity and models between the domains. The business requirements that are specified in the business processes must be properly analysed to elicit and specify functional requirements from them. Therefore, it is necessary to find means that help system analysts to adequately determine the system support to business processes.

This paper presents a RE approach (referred to as new approach hereafter) that provides detailed methodological guidance to elicit and specify the functional requirements of an IS from the business processes of an organization. The approach is based on the extension of BPMN (Business Process Modeling Notation) [19], and its purpose is to help system analysts specify the set of task descriptions that precisely and homogeneously represent the functional requirements of an IS.

The new approach is the result of a collaborative project with the software development company CARE Technologies (http://www.care-t.com), and practical experience using it is also presented. It has been used in several small/medium-size organizations, with which field trials have been carried out in order to acquire knowledge and to show that the approach can be used in practice. The new approach is based on another RE approach [8][9] (referred to as wider approach hereafter) that encompasses business process-centred organizational modelling, system purpose analysis, business process reengineering (considered as improvement), and requirements specification. The new approach is the result of practical experience with the wider one, from which improvements in requirements elicitation and specification were identified and several lessons were learnt, as explained below.

The new approach starts from the modelling in BPMN of the business processes that an organization wants to execute. The business process diagrams (BPD) are analysed and graphically enriched in collaboration with the stakeholders for the elicitation of functional requirements. Finally, these requirements are specified by means of task descriptions in textual templates, and are agreed upon with the stakeholders. The content of the templates is derived by following a set of guidelines that are based on the structure and automation of the business processes, and a criterion is used so that the granularity of task descriptions is homogeneous.

We are aware that the new approach might be considered not to be very original. Related work abounds and the problems that are addressed are not new. Nonetheless, we think that the value (i.e. originality) of the new approach is that it deals with important issues that are not properly addressed by other approaches. It is essential to understand, properly analyse and try to improve the business processes of an organization during the RE process, to precisely and homogeneously specify the functional requirements of an IS from them, and to promote stakeholder involvement. Furthermore, the new approach has been defined from practical experience.

The paper is organized as follows: section 2 presents the new approach; sections 3 revises related work; section 4 describes practical experience using the approach; finally, section 5 presents our conclusions and future work.

2 Description of the New Approach

As a result of the practical experience with the wider approach, its methodological guidance to elicit and specify task descriptions from BPDs has been refined and improved progressively. New patterns of system support to business processes have been discovered, new elements (concepts) have been introduced to analyse these patterns, the correspondence between the patterns and the functional requirements of an IS has been determined, and a significance criterion for functional requirements has been defined so that the granularity of task descriptions is homogenous.

The new approach (Fig. 1) consists of two stages: BPD enrichment and specification of task descriptions.

In the first stage, the "to-be" BPDs of an organization are labelled according to the system support that they will have, and the labelling is agreed upon with stakeholders. Next, the flow objects that are executed consecutively are identified in collaboration with stakeholders, who validate the identification. The output of this stage is the set of enriched BPDs of the organization.

Afterwards, textual templates are filled in order to specify the set of task descriptions that represents the functional requirements of an IS. This activity is carried out from the enriched BPDs by following a set of guidelines that determine the correspondence between BPD elements and the content of the textual templates. Part of the content must be agreed upon with stakeholders, and they must check the templates to validate the requirements.

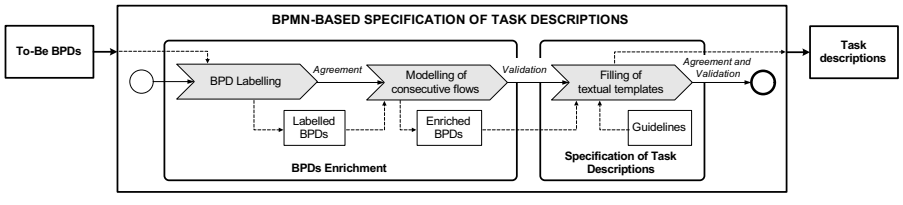


Fig. 1. New approach for the specification of task descriptions

2.1 BPDs Enrichment

Before the first stage is developed, the business processes that an organization wants to execute to fulfil its needs are modelled, and stakeholders collaborate in the process, as described in [8] and [9]. BPDs are completed with the textual specification of the business rules that have not been modelled graphically and a table that specifies the input and output data objects of the business tasks and their states. All business rules and data objects are not always represented graphically to facilitate the understanding of the BPDs.

BPDs, which represent business requirements, are then analysed and graphically enriched for the elicitation of functional requirements. System analysts have to precisely determine the system support that the business processes will have and the execution order of its flow objects, and they do it in collaboration with the stakeholders. These activities are explained in the following subsections.

2.1.1 BPD Labelling

In this activity, system analysts and stakeholders agree upon the automation of the business processes. BPMN tasks, events with triggers, and gateways that depict decisions are labelled according to the system support that they will have. The labels are: "O" (out of the system), if the flow object will not be part of the IS; "IS" (controlled by the system), if the IS will be in charge of the execution of the flow object with no human participation; or "U" (controlled by a user), if the flow object will be executed by a person that interacts with the IS.

On the basis of our experience, the semantics of the flow objects that will be out of the system is clear, but the semantics of those that will be controlled by the system or by a user might be confusing. Depending on their labels, an event happening will be thrown or caught by the IS or by a user that interacts with the system, the fulfilment of a gateway condition will be checked by the IS or by a user, and a task will be executed by the IS or by a user. In the latter case, the system will also take part in the execution of the task, but it will be executed because of the user's initiative.

In addition, system analysts and stakeholders agree upon the business rules and data objects that will be part of the IS.

2.1.2 Modelling of Consecutive Flows

In the second activity of the first stage, labelled BPDs are enriched by specifying those sequence flows that are consecutive flows, i.e. those sequence flows that link two flow objects that are executed one after another without an interruption. The graphical representation of a consecutive flow is an arrow with two arrowheads.

This type of connection does not exist in BPMN. The purpose of its definition is to be able to graphically represent the fact that two flow objects are always executed consecutively. As is explained below, homogeneity of task descriptions is based on the analysis of consecutive flows.

The identification of consecutive flows is carried out as follows. For each sequence flow of a BPD, system analysts have to determine if the target flow object is always executed immediately after the source flow object when a token is in the sequence flow. If so, both flow objects are linked by means of a consecutive flow.

Stakeholders' participation is essential in this activity. Stakeholders are the source of information from which the execution order of the BPD elements is modelled, and they must validate that the consecutive flows have been properly modelled according to how the organization executes or wants to execute its business processes.

An example of enriched BPD is shown in Fig. 2. It corresponds to a fragment of the real business process "item acquisition" of the Library and Scientific Documentation Office at Universidad Politécnica de Valencia. Due to page limitations, the complete BPD is not modelled, the business process is not described in detail, and the business rules that are specified textually and the table with the input and output data

objects are not shown. Anyway, we consider that the fragment is easy to understand, and the lack of this information does not hinder the understanding of the new approach. The complete BPD and its description can be downloaded from http://www.upv.es/entidades/ABDC/menu_634945c.html.

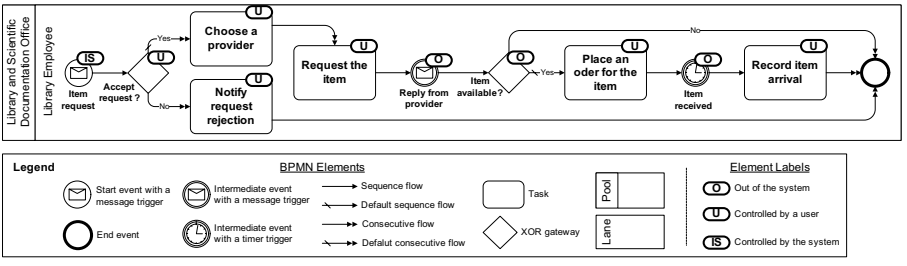


Fig. 2. Example of enriched BPD

2.2 Specification of Task Descriptions

In the second stage of the approach, the functional requirements of an IS are elicited and specified from the enriched BPDs. Functional requirements are specified by means of task descriptions in a textual template whose purpose is to specify adequate system support for business tasks. The content of the template is based on task & support descriptions [18] and essential use cases [5], but with some slight differences.

Unlike the wider approach, the granularity of business tasks modelled in BPMN and task descriptions may not be the same in the new approach. A task description can support several business tasks (subtasks of the task description), and the determination of these subtasks will be based on the consecutive flows of the enriched BPDs. A task description specifies IS support for the execution of a set of consecutive flow objects of an enriched BPD. The set includes flow objects that represent the subtasks of the task description, and the subtasks will be executed by the same user role and/or by the system.

Another difference with the wider approach is the definition of a significance criterion for functional requirements (task descriptions) so that their granularity is homogeneous and, thus, their specification is consistent and proper. The criterion is as follows: a task description is significant if no other task description is always executed immediately before or after the first one is executed. If there were two task descriptions that hindered the fulfilment of this criterion, both task descriptions would represent the same functional requirement in conjunction, so they should be specified in the same textual template. This criterion is a result of using analysis of consecutive flows for elicitation of task descriptions.

The sections of the textual template of a task description and the guidelines to fill them are presented in the following subsection. The sections either are derived from an enriched BPD or must be agreed upon with stakeholders. Table 1 shows an example of textual template (specified from the enriched BPD shown in Fig. 2).

Table 1. Example of task description

Task Description: **MANAGE ITEM REQUEST**			
Business Process: Item acquisition		**Role**: Library Employee	
Subtasks: Decide whether accept a request or not, Choose a provider, Request the item, Notify request rejection			
Triggers: Item request **Preconditions**: - **Postconditions**: -			
Input		**Output**	
Data Object	**State**	**Data Object**	**State**
Item Request	New	Item request	Accepted or Rejected
Provider	-		
Business Rules • The period to manage an item request depends on the type of item (books: 7 working days; films: 30 working days; magazines and journals: before 30th September) • An item is accepted or rejected on the basis of its existing copies, loans, and the explanation of its acquisition need • A provider is chosen on the basis of its delivery time, cost and complementary services			
User Intention		**System Responsibility**	
Normal			
2. Accept the request 4. Choose a provider 5. Request the item		1. Show item request details 3. Show the providers 6. Store the information	
Alternatives			
2.a.1. Reject the request 2.a.3. Notify the rejection (→End)		2.a.2. Store the rejection	
Extensions			
-		-	

2.2.1 Filling of Textual Templates

Several guidelines have been defined to fill the textual template of a task description. The guidelines specify what BPD elements correspond to the content of a task description, and there is a guideline to fill each section of a textual template. We are aware that some guidelines might be difficult to understand without an example, but it is not possible to exemplify each guideline due to page limitations. We are also aware that some parts of the explanation of the guidelines may be intuitive or well-known for some readers, but we have dealt with cases in which the understanding was not as simple as expected and led to misinterpretation.

Name

If a task description has only a subtask, then its name is the same as the name of the subtask. Otherwise, the name must be agreed upon with the stakeholders.

Business Process

The business process of a task description corresponds to the name of the enriched BPD from which the content of the task description is specified.

Role

The role of a task description is the system if all its subtasks are controlled by the system. Otherwise, the role is the participant in the business process of the task description whose lane contains the flow objects from which the subtasks of the task description that will be controlled by a user have been specified.

Subtasks

The subtasks of a task description are the flow objects that denote subtasks that will be supported by the IS. These flow objects are the BPD tasks that will be controlled by the IS or a user, and the gateways and "throwing" events that will be controlled by a user.

Triggers

The triggers of a task description are the flow objects that precede the first of its subtasks, cause the need to execute the task description, and will be controlled by the IS. These flow objects are the events that will be controlled by the system, and the gateways that will be controlled by the system if the role is the system.

Preconditions

The preconditions of a task description are the flow objects that precede the first of its subtasks and denote conditions that must be fulfilled before the task description can be executed and will be controlled by the IS. These flow objects are the "catching" events that will be controlled by a user, and the gateways that will be controlled by the system if the role is not the system.

Postconditions

The postconditions of a task description are the flow objects that follow the last of its subtasks and denote conditions that must be fulfilled after the task description is executed and will be controlled by the IS. These flow objects are the "throwing" events that will be controlled by the system, and the gateways that will be controlled by the system and can make the task description iterate.

Triggers, preconditions and postconditions can be combined conjunctively and disjunctively.

Input

The input of a task description is the data objects that are input of the flow objects from which its subtasks were defined and will be part of the IS.

Output

The output of a task description is the data objects that are output of the flow objects from which its subtasks were defined and will be part of the IS.

Business Rules

The business rules of a task description are the business rules that were specified textually in its business process, will be part of the IS, and affect the execution of the task description. They must be agreed upon with the stakeholders.

Normal (User Intention and System Responsibility)

The normal user intention and system responsibility of a task description are the set of actions that a user and the system carries out during the normal execution of the task description (set of subtasks of the task description that are always executed in its business process, or that are the default flow of a gateway and the branches that follow the gateway are not always executed in the business process of the task description).

Normal user intention and system responsibility must be agreed upon with the stakeholders, and their actions are jointly ordered according to their execution.

Alternatives

The alternatives of a task description are the set of actions that a user and/or the system may carry out when executing it, are not part of the normal execution of the task description, and are an alternative to the actions of normal user intention and/or system responsibility (i.e. actions that imply that some action of normal user intention and/or system responsibility is not executed and substitute it).

The actions of alternatives must be agreed upon with the stakeholders. They are ordered by means of three components: the same number as the first action that substitute; a letter to distinguish among alternatives; and another number to order the actions. The action of normal user intention or system responsibility that follows the last action of an alternative is put in brackets.

Extensions

The extensions of a task description are the set of actions that a user and/or the system may carry out when executing it, are not part of the normal execution of the task descriptions, and are an extension to the actions of normal user intention and/or system responsibility (i.e. additional actions that may be executed but do not imply that some action of normal user intention and/or system responsibility is not executed).

The actions of extensions must be agreed upon with the stakeholders. They are ordered like alternatives, but their first number corresponds to the action of normal user intention or system responsibility that precedes its actions.

3 Related Work

Business process modelling is part of most of the organizational modelling-based RE approaches for IS development, such as EKD [3], ARIS [22] and the UML-based approaches (e.g. [12]). There exist more approaches, but we consider these approaches to be sufficient to illustrate related work.

Some weaknesses these approaches are that they lack precise guidance for requirements elicitation and specification (they state what must be elicited or specified, but they do not explain in detail how to do it) and their specification of functional requirements might be considered inadequate because they just use a use case diagram or other simple model and do not provide guidelines for homogeneous granularity. In addition, ARIS and the UML-based approaches lack focus on business process improvement, and the diagrams of the UML-based approaches might be difficult for stakeholders to use and understand [23].

With regard to other RE approaches, scenario-based approaches [2] can be considered to be close to our approach. On the one hand, their purpose is to find possible ways to use a system, which is a similar purpose to normal user intention and system responsibility, alternatives and extensions. On the other hand, scenarios usually use the language of the application domain, facilitate agreement about system support to the business processes, and inter-relate system functionality and business processes [27]. As other authors [17], we think that the main weakness of scenario-based approaches is that they do not precisely explain how to obtain the scenarios and where their content comes from.

The original version of Lauesen´s task descriptions [18] and the new approach shares two main principles: specify adequate support for business tasks and delay the splitting of the work between the system and the users. Lauesen's task descriptions can be considered similar or equivalent to "to-be" BPDs of our approach, and his task & support descriptions to our tasks descriptions. The main differences are that the new approach uses graphical models for requirements elicitation, specifies system support on the basis of essential use cases, differentiates between alternatives and extensions, and extends the content of the original textual template with information such as business rules and input and output.

With regard to the definition of a criterion for homogeneous granularity and significance of functional requirements, this issue is missing in most of RE approaches. Nonetheless, there exist well-known approaches that provide criteria in order to assure that functional requirements are significant (such as Cockburn's goals [4] or

Lauesen's closure and "coffee break test" [18]). As other approaches (e.g. [13]), the difference of the new approach is that its guidance for homogeneity is more detailed, so we think that it facilitates analysts' work.

4 Practical Experience

This section presents the practical experience using the new approach with CARE Technologies and explains the lessons learnt.

4.1 Motivation of the Project

The collaborative project with CARE started in 2005, and its purpose is to link business and software domains in order to solve problems that the company has experienced and are related to the RE process. These problems arise when system analysts are inexperienced, they model large or complex systems, or the organization for which an IS is going to be developed is part of a domain with which they have not previously dealt.

The company uses OO-Method [20], a methodology for automatic software generation based on data-centred conceptual modelling. Among other advantages, this methodology can decrease development time and increase productivity. However, these advantages might disappear if problems related to the RE process arise. System analysts might have difficulty in modelling systems, and systems are sometimes deployed later than expected.

4.2 RE Practices in CARE Technologies

After analyzing its requirements practices, we argued that the company did not properly address business and system purpose understanding, communication with stakeholders, and, therefore, requirements elicitation, specification and validation. Furthermore, problems in subsequent development stages may occur.

The conceptual schemas of OO-Method consist mainly of a class diagram that is enriched with information about methods execution. The class diagram is the main system model, and analysts just provide some unstructured textual descriptions about the requirements and validate them on the class diagram or on the application. Senior analysts feel comfortable with this technique, but we think it should be improved:

- Class diagrams alone might not be appropriate for communicating and validating requirements, there are few studies addressing the ability of stakeholders to understand class models, and they can be complex for people that have not been trained in object-oriented modelling [10]. In addition, objects might not be a good way of thinking about an application domain [26].
- Requirements validation should be carried out with reference to organizational concerns instead of with reference to system functionality [21]. Furthermore, requirements validation on the generated applications causes the late detection of errors, thus their correction might be much more expensive than if errors had been detected in earlier development stages.

– Maintenance might be complex for systems whose analysts do not longer work for CARE. System documentation is usually scarce and inadequate, so system modification can be very difficult for new analysts of a system when they just have a class diagram and the application to understand the system and its requirements.

CARE has previously tried to solve its requirements problems, and three approaches have been developed as a result of the collaboration between the company and Universidad Politécnica de Valencia. However, the approaches have weaknesses and most of CARE analysts think that the benefits of their use against their time cost are not worth, so the company thinks that they do not fit its needs:

– Modelling of use cases and sequence diagrams [16]: this approach is solution-oriented, so it does not properly analyse application domain; in addition, its diagrams might be difficult for stakeholders to validate, and they are too detailed (the operations that are specified are equivalent to class methods in many cases).
– Modelling with the i* framework [14]: as acknowledged by the proponents of the approach [15], i* models need to improve features such as granularity and refinement; furthermore, they might be too complex for stakeholders to understand and for non-experienced analysts to use.
– Analysis of business processes, goal trees and use cases [7]: goal trees might be too complex and thus hard to use and understand, and detailed guidance to bridge the gap between use cases and the other models is necessary; nonetheless, this approach can be regarded as the direct antecedent of the wider approach.

4.3 Application of the Approach

A very positive point of collaborating with CARE is that it belongs to a holding company (http://www.olivanova.com). As a result, other organizations of this company have been used to evaluate the new approach. CARE had developed software for the organizations previously, so the technique they usually use, the RE approaches of the previous subsection and our approach could be compared.

As mentioned above, the wider approach and the new one have been applied in field trials to acquire knowledge and to show that they can be used in practice. Both stakeholders of the organizations (managers, employees and end-users) and CARE analysts participated. First, we held meetings with stakeholders to model the organizations. They described the activity of the organizations, and organizational documentation was compiled. BPDs were then modelled and stakeholders validated the diagrams. Several iterations were needed to obtain the final version. Next, analysis of system purpose was carried out to find out means to meet systems goals and determine how they affected business processes. BPDs were labelled, and functional requirements were specified and validated. More details about application of the wider approach can be found in [8] and [9].

4.4 Lessons Learnt

This subsection summarizes and discusses the main lessons that have been learnt during practical experience and are related to the new approach. Some of them could be used for other RE approaches (lessons 1, 2, 3, 4, 5, 6 and 7), and some of them are specific to the new approach (lessons 8, 9 and 10).

1) Detailed methodological guidance is essential for the use of a RE approach

Lack of detailed methodological guidance has been one of the main problems of the RE approaches that CARE has tried to use (both the approaches of subsection 4.2 and other existing approaches). The approaches explain what information has to be compiled, but system analysts have problems when using them because they do not know exactly how to compile it.

This problem has been addressed in the new approach. Detailed guidance has been provided so that analysts do not get confused when using the approach and this use is as straightforward as possible.

2) BPMN facilitates communication

One of the first facts that CARE requested to be proved was the claim that BPMN is easy to use and understand and, thus, facilitates communication. Communication was addressed from two viewpoints: communication between analysts and stakeholders, and communication among analysts.

When modelling the business processes of the organization, stakeholders get used to BPMN even quicker than we expected. After one or two meetings, they were able to completely understand the BPDs (although it must be said that we have not used the complete notation in any field trial), and they even fixed modelling errors.

With regard to the communication among analysts, analysts that were not expert in BPMN were asked to interpret BPDs of organizations that they did not know in depth. The result was that they understood them easily most of times, and they just had problems when documentation of the organizational activity (business rules, data objects…) was insufficient.

3) Notation extension and graphical representation can facilitate understanding

When proposing to extend BPMN graphically with labels for flow objects and with consecutive flow, some analysts didn't like the idea. They wondered if more notation elements were really necessary given that BPMN is quite extensive, and they asked if there were no mechanisms in BPMN to specify the automation of business processes. These mechanisms exist in the form of attributes, but just for BPMN tasks.

Since we argued that labelling and modelling of consecutive flows were necessary for the new approach, analyst finally agreed upon their use. Furthermore, after using these elements, they acknowledged that these graphical elements helped them to better understand the BPDs, and thus BPMN expressiveness was improved.

Stakeholders also liked the graphical extensions. Understanding of the proposed solution (system support) for their business processes was easy, much higher than if new graphical elements had not been used (i.e. using or defining attributes of the BPMN elements).

Nonetheless, we acknowledge that other mechanisms could have been adopted, and some of them were analysed. For example, instead of defining consecutive flow, BPMN sub-process could have been used for encapsulation of consecutive flow objects. However, we finally decided to use sub-processes just as mechanism for refinement and decomposition of business processes without imposing any constraint to the execution order of their flow objects, as BPMN does.

4) There exists a semantic gap between BPMN and functional requirements

The existence of a semantic gap between business and system domains that must be bridged has been acknowledged commonly, and this gap has become evident for us as a result of using BPMN for the elicitation of functional requirements.

When BPMN is used from a business perspective (it might also be used from a system perspective), the semantics of its elements is different from the semantics of the elements of a task description. This difference is clear if the guidelines for filling the textual template of a task template are analysed. For example, depending on the system support to them, a BPMN event with a trigger may not correspond to the trigger of a task description, and the subtasks of a task description do not only correspond to BPMN tasks.

5) Granularities of business tasks and of functional requirements can be different, and requirements homogeneity is essential for specification consistency

We think that the main refinement in the wider approach has been the differentiation between the granularities of business tasks and of task descriptions. They were the same initially, but we realised that granularities could be different and task descriptions would be inconsistently specified if their granularities were not homogeneous. Using Table 1 as example, we consider that it would be nonsense to specify four task descriptions (one per subtask) and, therefore, four functional requirements from it. Since the subtasks are always executed consecutively, we think that their specification as functional requirements is significant as a whole (an only requirement), and not as separate requirements. As a result of the granularity change, the alternatives and extensions sections of the textual templates were included. In addition, using the initial granularity, the number of functional requirements (task descriptions) was higher, so more difficult to be managed.

However, a problem arose: how detailed should then a business task be? On the one hand, granularity should allow task descriptions to be homogenously specified. On the other, stakeholders should feel comfortable with granularity so that they thought that BPDs were detailed enough to properly depict the organizational activity. Finally, we decided that business tasks should be as detailed as stakeholders desired (felt comfortable with). Using Fig. 2 as example, a stakeholder might feel comfortable if the business tasks “Choose a provider” and “Request the item” were modelled as an only task, but another might not.

Afterwards, we conceived the concept of consecutive flow and defined the significance criterion as mechanisms for homogeneous specification of task descriptions. Evidently, depending on the granularity of business tasks, task descriptions will support more or fewer subtasks, and less or more interaction between a user and a system will have to be discovered.

6) The use of a RE approach will depend on the characteristics of the projects

When asking analysts about the usefulness of the new approach, we obtained different opinions. Although all of them stated that the approach allowed them to better understand the requirements, there were some senior analysts who did not think that the approach could improve their job significantly and would probably not use it.

These analysts are very skilled in using OO-Method and interacting with customers, and they usually model a system while the customer describes what the system should do, so they can quickly generate it, validate it, and fix it if needed. However, most of the junior analysts, who have less experience in dealing with customers and, therefore, in understanding what is needed, considered that the new approach could really help them.

We think that these comments are a reflection of common practices in IS development. Models are only used when they are believed to be useful [6]. In our case, some

senior analysts do not think that the new approach can accelerate their job or improve the final system, whereas junior analysts think that it can improve their performance.

In summary, the use of our approach by system analysts will depend on their experience and knowledge about the application domain, and on system complexity.

7) It is essential to link RE approaches to subsequent development stages

Although we consider that the wider approach and the new one can solve the problems that have been initially addressed in the project, they must be linked to OO-Method to be useful for CARE. A RE approach is not useful per se, but it must be appropriate for the software development process into which it is integrated [25].

Furthermore, this link will improve the approaches. They are mainly centred on the analysis of the behaviour (business processes) of an organization, but they must also address its data. Thus, data modelling will be carried out on the basis of OO-Method.

8) Alternatives of task modelling in BPMN do not affect task descriptions

When using BPMN, a business task can be modelled in different ways. For example, the gateway "accept request" shown in Fig. 2 could have been modelled as a BPMN task depending on the preferences of system analysts or stakeholders. However, this fact does not affect the specification of task descriptions. Unlike the existing guidance of the wider approach, the existence of alternative ways of modelling business tasks (which will be subtasks of task descriptions) in BPMN has been analysed and included in the guidelines of the new approach.

9) The filling of the textual templates of the task descriptions is variable

The content of the textual templates of the task descriptions depends on the BPDs. Their content comes from them, and their correspondence is determined by guidelines. However, BPD modelling is variable, and therefore the textual templates also are. For example, a business rule might be modelled: 1) as a gateway; or 2) textually as documentation of a BPD. Therefore, this business rule would correspond to: 1) to a trigger, precondition, postcondition or a place that originates an alternative or an extension; or 2) to a business rule of a textual template.

However, we do not think that variability of BPD modelling and, thus, of the content of the textual templates is a weakness or problem of the new approach. What is important for us is that all the organizational concerns that will affect or will be controlled by an IS (e.g. business rules) are specified in the task descriptions, regardless of where they are specified.

Furthermore, guidelines advise analysts how to fill the textual templates, but do not impose a unique way of filling. We have presented the set of guidelines that we like and we have agreed upon with CARE, but we are aware that other authors could like other guidelines. For example, other authors could prefer not to specify input and output data objects and to consider them as pre and postconditions, respectively.

10) The practical experience has limitations

The practical experience that has been presented has several limitations. Apart from the limitations reported in [8] and [9] for the wider approach (use in more and larger projects, and formal surveys and experiments), the new approach must be used in projects in which a legacy system exists so that the approach is evaluated in this situation. We are considering the options of including the description of legacy systems as part of organizational modelling and of defining a new label for BPD elements ("L", controlled by a legacy systems), but they must be put into practice in order to evaluate them and find out the effect on the current methodological guidance.

5 Conclusions and Future Work

This paper has presented an approach that provides detailed methodological guidance in order to properly elicit and specify task descriptions for an IS from business processes modelled in BPMN. The approach represents a deep refinement and improvement of previous guidance thanks to its use with a software development company, and details about practical experience and lessons learnt have been presented.

The approach has analysed the use of BPMN for the specification of task descriptions. As a result, the gap between BPMN and task descriptions has been bridged, BPMN has been extended graphically by specifying the automation of its elements with labels and by defining the concept of consecutive flow, and BPMN expressiveness has been improved. Task descriptions are specified in a standard template from the analysis of BPDs thanks to a set of guidelines that determines the correspondence between the business processes and the system support that they need, and a significance criterion for task descriptions has been presented to specify them homogeneously.

In addition, stakeholders collaborate in developing the activities of the approach. They participate actively by validating models, and by agreeing upon the decisions on the automation of business processes and on the specification of functional requirements with system analysts.

Apart from the improvements that have been pointed out above (link to OO-Method, use of the approach with legacy systems …), in our future work we want to develop a tool that automates the use of the approach and introduce a technique for the analysis of system non-functional requirements. We also want to extend the textual template of the task descriptions in order to derive abstract user interfaces that facilitate requirements validation. Finally, since the approach is being and will be applied in new projects, refinements and improvements might be found.

Acknowledgements. The authors would like to thank Anne Person, Martin Glinz, Patrick Heymans and the reviewers for their useful comments and suggestions for the final version of the paper.

References

1. Alexander, I., Bider, I., Regev, G. (eds.): REBPS 2003: Motivation, Objectives and Overview. Message from the Workshop Organizers. REBPS 2003, CAiSE 2003 Workshops, Klagenfurt/Velden, Austria (2003)
2. Alexander, I., Maiden, N.: Scenarios, Stories, Use Cases. John Wiley and Sons, Chichester (2004)
3. Bubenko, J., Persson, A., Stirna, J.: EKD User Guide (2001), `http://www.dsv.su.se/~js`
4. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley, Reading (2001)
5. Constantine, L., Lockwood, L.: Software for Use. Addison-Wesley, Reading (1999)
6. Davis, I., et al.: How do practitioners use conceptual modelling in practice? Data & Knowledge Engineering 58(3), 359–380 (2006)
7. de la Vara, J.L., Sánchez, J.: Business process-driven requirements engineering: a goal-based approach. In: BPMDS 2007, Trondheim, Norway (2007)

8. de la Vara, J.L., Sánchez, J.: Improving Requirements Analysis through Business Process Modelling: A Participative Approach. In: Abramowicz, W., Fensel, D. (eds.) BIS 2008, Innsbruck, Austria. LNBIP, vol. 7, pp. 167–178. Springer, Heidelberg (2008)
9. de la Vara, J.L., Sánchez, J., Pastor, Ó.: Business Process Modelling and Purpose Analysis for Requirements Analysis of Information Systems. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 213–227. Springer, Heidelberg (2008)
10. Dobing, B., Parsons, J.: Understanding the role of use cases in UML: a review and research agenda. Journal of Database Management 11(4), 28–36 (2000)
11. Dumas, M., van der Aalst, W., ter Hofstede, A.: Process-Aware Information Systems. Wiley, Chichester (2005)
12. Eriksson, H., Penker, M.: Business Modeling with UML. John Wiley and Sons, Chichester (2000)
13. España, S., González, A., Pastor, Ó.: Communication Analysis: a Requirements Elicitation Approach for Information Systems. In: CAiSE 2009, Amsterdam, Netherlands (2009)
14. Estrada, H., Martínez, A., Pastor, Ó.: Goal-Based Business Modeling Oriented towards Late Requirements Generation. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 277–290. Springer, Heidelberg (2003)
15. Estrada, H., et al.: An Empirical Evaluation of the i* Framework in a Model-Based Software Generation Environment. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 513–527. Springer, Heidelberg (2006)
16. Insfran, E., Pastor, Ó., Wieringa, R.: Requirements Engineering-Based Conceptual Modelling. Requirements Engineering 7(2), 61–72 (2002)
17. Jones, S., et al.: Informing the Specification of a Large-Scale Socio-technical System with Models of Human Activity. In: Sawyer, P., Paech, B., Heymans, P. (eds.) REFSQ 2007. LNCS, vol. 4542, pp. 175–189. Springer, Heidelberg (2007)
18. Lauesen, S.: Task Descriptions as Functional Requirements. IEEE Software 20(2), 58–65 (2003)
19. OMG: Business Process Modeling Notation (BPMN) Specification v.1.1 (2008), http://www.bpmn.org
20. Pastor, O., Molina, J.C.: Model-Driven Architecture in Practice. Springer, Heidelberg (2007)
21. Rolland, C., Prakash, N.: From conceptual modelling to requirements engineering. Annals of Software Engineering 10(1-4), 151–176 (2000)
22. Scheer, A.W.: Aris - Business Process Modeling, 3rd edn. Springer, Heidelberg (2000)
23. Siau, K., Cao, Q.: Unified modeling language – a complexity analysis. Journal of Database Management 12(1), 26–34 (2001)
24. Sommerville, I., Sawyer, P.: Requirements Engineering: A Good Practice Guide. Wiley, Chichester (1997)
25. Verner, J., et al.: Requirements engineering and software project success: an industrial survey in Australia and the U.S. Australasian Journal of Information Systems 13(1) (2005)
26. Vessey, I., Coner, S.: Requirements Specification: Learning Objects, Process and Data Methodologies. Communications of the ACM 37(5), 102–113 (1994)
27. Weidenhaupt, K., et al.: Scenarios in System Development: Current Practice. IEEE Software 15(2), 34–45 (1998)
28. Zave, P., Jackson, M.: Four Dark Corners of Requirements Engineering. ACM Transactions on Software Engineering and Methodology 6(1), 1–30 (1997)

Clarifying Non-functional Requirements to Improve User Acceptance – Experience at Siemens

Christoph Marhold[1], Clotilde Rohleder[1,2], Camille Salinesi[2], and Joerg Doerr[3]

[1] Siemens PL (DE) GmbH, Lina-Ammon-Strasse 22, D-90471 Nuremberg
[2] CRI - Université Paris 1, 90, rue de Tolbiac, F-75013 Paris
[3] Fraunhofer Institute Experimental Software Engineering, D-67663 Kaiserslautern
{christoph.marhold,clotilde.rohleder}@siemens.com,
camille.salinesi@univ-paris1.fr, joerg.doerr@iese.fraunhofer.de

Abstract. [Context and motivation] The starting point for software development is usually the system requirements. The requirements, especially non-functional requirements specified in a document are often incomplete and inconsistent with the initial user needs and expectations. **[Question/problem]** Experience at Siemens showed us that programmers working on software development often have trouble interpreting under-specified non-functional requirements, resulting in code that does not meet the users' quality expectations and contains "quality faults" that can only be detected later through expensive user acceptance testing activities. **[Principal ideas/results]** In this problem statement paper, we investigate the need for clarifying non-functional requirements in software specifications to improve user acceptance. In particular we focus on establishing the role of non-functional requirements on user acceptance. **[Contribution]** Our contribution is that we emphasize the need for a systematic empirical study in this area. We propose a possible set-up where a number of hypotheses have been developed that a systematic experiment will help to validate. Our work is based on industrial experiments at Siemens, in the particular context of the installation of a Product Lifecycle Management (PLM) system.

Keywords: User Acceptance, Non-Functional Requirements.

1 Introduction

It is crucial for implementers that the systems they develop are finally accepted by their users. New systems must therefore meet explicit and implicit criteria for acceptance. Experience shows that user non-acceptance originates more often from inadequate non-functional requirements (NFR) more than from problems with functional requirements [6]. There are many NFR-related causes of user non-satisfaction: for example ambiguous, incomplete, wishful thinking, inconsistent, unusable, silent, under-specified or under-sized NFR. From users' point of view, NFRs are qualities that the system must show. In practice, users are unsatisfied because the system does not achieve their expectation in terms of quality: the

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 139–146, 2009.

perceived quality does not match the expected quality. Generally errors detected during user acceptance tests are reported as bugs. But in fact, non-acceptance can arise from many other issues than bugs. In this paper, we refer to quality as defined in the ISO9126 standard [13] that classifies NFRs as Software Quality tree (Fig. 1), and distinguishes between the internal/external qualities of systems, and quality in use.

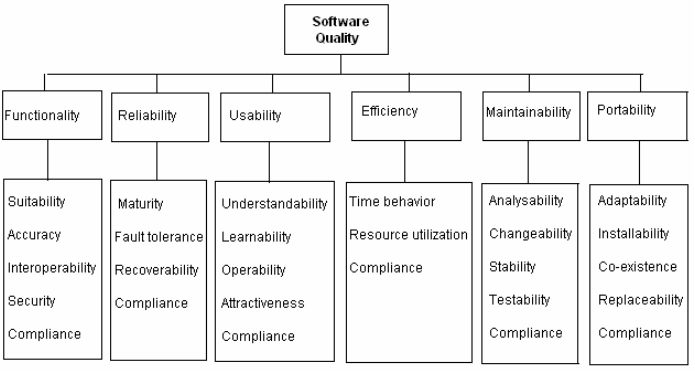


Fig. 1. ISO9126 Software Quality Tree

One major hypothesis underlying our work is that the latter is influenced by the former. Our vision is to significantly increase user acceptance of complex systems in better implementating quality goals (in the shape of an NFR). To our knowledge, there is surprisingly little research to date on the mechanisms, technologies and attitudes that address the impact of non-functional requirements on user acceptance. We have therefore decided to undertake a research work on this topic.

This paper is the first of a series of investigations of the impact on user acceptance of a methodology for clarification of non-functional requirements using an empirical approach. The paper has two parts: In the first part, we introduce the domain of PLM, and report Siemens's experience and view about critical factors of user acceptance while installing PLM systems, and about the perception of the importance of NFR in these projects. The second part of the paper uses this analysis of past experience at Siemens to draw the basis for a series of systematic experiements about the correlation between NFRs and user acceptance. The originality of the paper is twofold (a) it is an industry experience based exploration of a fundamental RE problem, and (b) it explores an issue that has received so far little attention in RE research. This paper reports early results from our analysis in the industrial context of PLM. In the long term, we hope to be able to develop a method that helps predict user acceptance for any kind of complex system based on an analysis of NFR specifications.

The paper is organized as follows: Section 2 describes the research problem in the PLM context. Section 3 presents the need for an empirical study. Related works and conclusions are given respectively in Section 4 and Section 5.

2 User Acceptance and NFR Specification in the PLM Context

2.1 Challenges of PLM Systems Acceptance by Users

Product lifecycle management (PLM) is "the process of managing the entire lifecycle of a product from its conception, through design and manufacture, to service and disposal. PLM integrates people, data, processes and business systems and provides a product information backbone for companies and their extended enterprise" [2]. Installing a PLM system implies –like with other complex COTS such as ERPs – some kind of matching between users' requirements and the requirements that the system is able to satisfy [20]. Fig. 2 shows that PLM is a strategic business approach that applies a consistent set of business solution in support of collaboration creation, management, dissemination, and use of product definition information across the extended enterprise from concept to end of life [2].

Fig. 2. Overview PLM Product Lifecycle Management [9]

The worldwide PLM applications market in 2007 amounted to $ 8.7 billion. There is no doubt that PLM is now broadly valued by large manufacturers as well as by small and medium-sized business. Cost, effort, time and complexity of implementing a PLM system can be compared to those for implementing an ERP system. Siemens's experience of implementing its PLM Teamcenter solution showed that the user acceptance tests had best results when non-functional requirements on response time or ease of use had been specified prior to implementation. PLM experts reported bad user acceptance in projects where the specification of non-functional requirements had been focused only on technical non-functional requirements or where user oriented non-functional requirements [14] had been insufficiently specified. Time and efforts spent in projects for correcting the final software product to improve user acceptance (projects of type A) compared to time and efforts spent in projects where more attention was paid to specifying sufficiently all NFRs to ensure user acceptance (projects of type B) was evaluated to be around 100. Moreover, we observed that the level of user acceptance reported in projects of type A has never reached the same level than the user acceptance reported in projects of type B. PLM experts at Siemens report that there is a correlation between specifying non-functional requirements and

improving user acceptance. Developers and technical project stakeholders with direct contact with customers are asking for more formality in NFR specification, whereas project managers responsible for budgets want less NFR formalization.

2.2 State of the Practice on NFR Specification for PLM System Installation

In practice, although one could assume that some care is taken when writing down the initial requirements including the NFRs while implementing such a complex system as a PLM, the role of a clear specification of the NFRs is usually undermined. Besides, user acceptance test documents tend to be little more than bug reports. We observed that in fact, it is often assumed that a lot of quality information is obvious at this time of the project or is agreed verbally. The main purpose of a PLM system is to enable collaboration among users. PLM systems handle a large collection of collaborative data (requirements specifications, simulation data, design 2D files, 3D models, bill of material, production plans, sales and marketing data, logistics, etc) from the early stage of product development until the maintenance phase. As a result, PLM systems have extremely diverse kinds of users: requirements engineers, CAD designers, CAE and CAM engineers, ERP users, maintenance technicians, etc. Each have specific expectations with respect to the PLM tool, not only in terms of functionality, but also in terms of ergonomy, performance, interoperability with other systems, and ability to support business goals. Besides, PLM systems must handle extremely different fields of application which results in extremely different NFR priorities [19]. For example in OEM Automotive Supplier sector, NFR "Portability" has "absolute" priority because of multi CAD systems (NX, ProE, CATIA, etc.) on different platform infrastructures (Windows, SUN, IBM, HP) used by key PLM users, whereas in the field of Defense the NFR "Security" comes first.

3 Empirical Investigation – Design and Result

3.1 Research Questions of Interest and Hypotheses

We decided to undertake an empirical investigation of the relationship between NFRs and user acceptance for several reasons. Besides (i) to assess the usefulness of formal NFR specification models, we are interested in (ii) evaluating whether meeting NFRs achieves better results than acceptance tests and (iii) determining the cost, in terms of extra efforts, to get user acceptance if NFRs are not adequately considered. Overall, the purpose of our empirical investigation is to determine the influence of NFR related factors on the acceptance by users of enterprise-wide systems such as PLM, ERP, etc. The investigation is split in two phases: an initial study, and a series of interviews. The initial study aims at exploring the relationship between NFR specifications and user acceptance in a qualitative way. The final result expected from this study is a series of hypotheses that can be quantitatively validated. This study is qualitative. It is mostly based on expert feedback and lessons learned from PLM installation experiences. The purpose of interviews is to validate quantitatively the hypotheses specified during the empirical study phase, and based on the results, to develop a systematic method that would help predict user acceptance from NFRs. Interviews during this second phase will be undertaken both with users and

developers of the PLM system. The questions that will be asked will be about their perception of system in relationship with NFRs, under the same perspective as defined in our hypotheses. Both the initial study and the questions raised during interviews have to be designed to deal with the following two research questions: RQ1: what are the non functional-related factors that influence user acceptance of enterprise-wide systems such as PML? RQ2: what are the influence flows from early requirements specifications down to user acceptance?

3.2 Possible Set-Up of an Empirical Study

We should be able to determine the correlation between the clarity of NFRs and user acceptance in setting a number of hypotheses that the interviews will help to validate. Based on the aforementioned return on experience, we drew a series of hypotheses of potential interest. For feasibility reasons, we reduced the list to 3 hypotheses as follows.

H1: Quality of non-functional requirements specification influences user acceptance

Experience: One customer specified following requirement concerning PLM Graphical User Interface: "GUI should be very simple and attractive for informational users". Developers needed to know if a complete new GUI had to be developed or if the current ones could be customized to be accepted by the informational users. The requirements had to be specified more precisely: how could the GUI be more attractive for the informational users? The quality of specifications of non-functional requirements seems to help programmers better understand the non-functional requirements in comparison with textual or semi structured descriptions.

H2: User acceptance increases when users are involved in the NFR priorization

Experience: in the preliminary phase of specifications –Prove of Concept- the users are committed in the weighting of non-functional requirements. In that phase, we pre-align the prioritizations of non-functional requirements according to the expectations.

H3: Improving user acceptance is not a continuous function of satisficing NFRs

Experience shows that user acceptance has a maximum level to be reached. There is a maximum limit of efforts that should be invested in satisficing NFRs to get maximized user acceptance. If the effort goes further, the user acceptance will never be better.

These hypotheses shall be checked in a systematic experiment. We need to compare efforts and time spent in getting the mile stones user acceptance test within different projects having different quality of specifications. We aim to work in test-driven development and search how much this emphasizes the contribution of testing and test cases, when their construction anticipates the actual development of the code. The strategy of empirical experiment is still in discussion. The advantage of performing case studies with focus groups would provide detailed feedback and valuable information. But we have to check if the results are able to be generalized to other PLM groups. If this study proves the importance of meeting the NFRs to improve the user acceptance, we should develop methods within PLM delivery methodology that advocates an expressive documentation form of the NFRs.

4 Related Works

Improving user acceptance is an important issue in industry. The issues met to install PLM systems may not be very different from the ones met to maintain an Information System, to personalize ERP or to create new market product. The literature proposes different research approaches to address user acceptance. Some approaches consist in surveys with the goal to assess system acceptance. For example, [16] defines a system acceptance indicator (SAI) and [18] and [5] propose similar approaches. The activity of interest [18] is the integration of an extant soft-system (human workforce) and the modified hard-system (IT system) [5]. Our approach is new because we adopt a requirement-driven point of view, whereas the aforementioned approaches focus on social and psychological and technical aspects such as HC interaction within system implementation. The importance of clarifying non-functional requirements is also a topic of interest in different areas of Requirements Engineering. For example, [10], [11] proposes to represent the NFRs as clouds. [15] used similar goal representation to visualize quality components. [3] [4] propose to represent the non-functional requirements by goals according to the decomposition methods of [3] and [1]. [14] focussed on user oriented non-functional requirements but did not mention user acceptance. Potts [21] used the term 'fitness for use' to designate the concept. 'Fitness' has been studied under different perspectives and was considered as a concept per se that can be modeled by [20]. Melnik's study [17] focused on the use of tables called Fit user acceptance tests for specifying functional requirements. Our approach is different because we are not starting from the user acceptance test to clarify functional requirements. We wish to get enough expertise to know how to specify requirements, in particular non-functional requirements, such that they will favourably impact user acceptance. We could also refer to methods like the AMUSE methods [6], [7], [8] where the results are not tailored to the medical domain or discussions on the nature of NFRs like in [12]. In [7] the authors show an approach that was designed to appraise and measure the users' (future) satisfaction in the requirements engineering phase. In this phase little systematic guidance exists on how to evaluate the effects of features on user satisfaction early on, and on how to contribute the results to the development process. The AMUSE approach [7], [8] claims to close this gap. It helps requirements engineers and product managers to select the most promising features, i.e., the ones that will satisfy the user the most, already in the requirements phase. The approach does not differ between functionality and non-functionality but deals with both in the general term feature. The approach uses a standardized user satisfaction measurement device (questionnaire), a feature appraisal and prioritization methodology, and some tool support. For evaluation of user acceptance, the standardized questionnaire could be used in the future to evaluate the effect of NFRs on user acceptance.

5 Conclusion

Experience with user acceptance of PLM system shows the need of meeting first the quality requirements. We believe that the user acceptance may depend on how well the NFRs are developed. We believe empirical research is necessary to evaluate the impact and efficiency of expressing NFRs in a more formal way. Also an in-depth

research is needed to increase understanding of the interplay between NFRs and user acceptance. We recommend the development of a process model (methodological framework) for development of NFRs. Another important issue is how to effectively guide the elicitation of non-functional requirements and check their correctness in view of user acceptance. The research should ensure maximum user acceptance of the PLM implementation.

References

1. Castro, J., Kolp, M., Mylopoulos, J.: Towards Requirements-Driven Software Development Methodology: The Tropos Project, Information Systems (2002)
2. CIMdata (2003), http://www.cimdata.com
3. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Boston (2000)
4. Chung, L., Nixon, B.A., Yu, E.: Dealing with Change: An approach Using Non-Functional Requirements. In: Proceedings of the Second Internation Symposium on Requirements Engineering, York, England. Requirements Engineering Journal, pp. 238–260. Springer, London (1996)
5. Checkland, P.: Systems Thinking, Systems Practice – Includes a 30 year retrospective. John Wiley and Sons, Chichester (1999)
6. Doerr, J., Kerkow, D., Landmann, D.: Supporting Requirements Engineering for Medical Products - Early Consideration of User-Perceived Quality. In: Association for Computing Machinery (ACM): 30th International Conference on Software Engineering. ICSE 2008, pp. 639–647. IEEE Computer Society, Los Alamitos (2008)
7. Doerr, J., Hartkopf, S., Kerkow, D., Landmann, D., Amthor, P.: Built-in User Satisfaction - Feature Appraisal and Prioritization with AMUSE. In: Sutcliffe, A., Jalote, P. (eds.) Proceedings of 15th IEEE International Requirements Engineering Conference, pp. 101–110. IEEE Computer Society, Los Alamitos (2007)
8. Doerr, J., Kerkow, D., Koenig, T., Olsson, T., Suzuki, T.: Non-Functional Requirements in Industry - Three Case Studies Adopting an Experience-based NFR Method. In: 13th IEEE International Requirements Engineering Conference (2005)
9. Eigner (2005), http://vpe.mv.uni-kl.de/cms/index.php?id=274
10. González-Baixauli, B., Sampaio do Prado Leite, J.C., Mylopoulos, J.: Visual Variability Analysis for Goal Models. In: Requirements Engineering Conference, pp. 198–207 (2004)
11. González-Baixauli, B., Laguna, M.A., Sampaio do Prado Leite, J.C.: A Meta-model to Support Visual Variability Analysis. In: First International Workshop on Variability Modelling of Software-intensive Systems, Limerick, Ireland (2007)
12. Glinz, M.: On Non-Functional Requirements. In: International Conference on RE (2007)
13. ISO/IEC 9126-1: Software Engineering - Product Quality - Part 1: Quality Model (2001)
14. Keller, R.K., Schauer, R.: Design Components: Towards Software Composition at the Design Level. In: Proceedings of International Conference on Software Engineering, Kyoto, Japan, pp. 302–311 (1990)
15. Lapouchnian, A., Yu, Y., Mylopoulos J., Liaskos S., Sampaio do Prado Leite J.C.: From stakeholder goals to high-variability software design, Tech. rep., University of Toronto (2005), http://ftp.cs.toronto.edu/csrg-technical-reports/509
16. Lehane, P., Huf, S.: Towards understanding system acceptance: the development of an assessment instrument and work practice. In: Proc. of OZCHI, Canberra, Australia (2005)

17. Melnik, G., Read, K., Maurer, F.: Suitability of fit user acceptance tests for specifying functional requirements: Developer perspective. In: Extreme programming and agile methods - XP/Agile Universe 2004, pp. 60–72 (2004)
18. Mwanza, D.: Towards an Activity-Oriented Design Method for HCI research and practice, Knowledge Media Institute, The Open University, Walton Hall, United Kingdom (2002)
19. Rohleder, C.: Visualizing the Impact of Non-Functional Requirements on Variants– A Case Study. In: International Conference on Requirements Engineering, REV 2008, Barcelona (2008)
20. Salinesi, C., Rolland, C.: Fitting Business Models to Systems Functionality Exploring the Fitness Relationship. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681. Springer, Heidelberg (2003)
21. Potts, C.: Fitness for Use: The System Quality that Matters Most. In: International Workshop on Requirements Engineering: Foundation of Software Quality, Barcelona (1997)

Scenarios in the Wild: Experiences with a Contextual Requirements Discovery Method

Norbert Seyff[1], Florian Graf[2], Neil Maiden[1], and Paul Grünbacher[2]

[1] City University London, Centre for HCI Design, London EC1V 0HB, UK
n.seyff@soi.city.ac.uk, n.a.m.maiden@city.ac.uk
[2] Johannes Kepler University, Systems Engineering and Automation, 4040 Linz, Austria
fg@sea.uni-linz.ac.at, pg@sea.uni-linz.ac.at

Abstract. [Context and motivation] A number of ethnographic approaches are available to gather requirements where they emerge, i.e. in the workplace of future system users. **[Question/problem]** Most of these approaches do not provide guidance and software tool support for on-site analysts. **[Principal ideas/results]** In this paper we present a tool-supported contextual method that combines key benefits of contextual inquiry and scenario-based techniques. It aims to improve guidance and support for on-site analysts performing a contextual requirements discovery. **[Contribution]** We applied this method in the Austrian Alps to discover stakeholder's requirements for a ski tour navigation system. This paper reports on this inquiry and analyses its results. Moreover, we discuss lessons learned and conclusions.

Keywords: Requirements elicitation, scenarios, contextual inquiry, mobile computing.

1 Introduction

Observing and interacting with people in their work context to study and learn about their needs and expectations is not new. Several approaches support contextual investigations [1], [2], [3]. Most of them are based on ethnography, which itself is anything but a unified method [4]. In recent years ethnographically-informed approaches have appeared in the field of human-computer interaction (HCI) and requirements engineering (RE) [3]. Researchers and practitioners have used ethnographic methods to support RE in various domains including air traffic control [5] and underground control rooms [6]. In some approaches ethnography was combined with existing RE approaches. For example, viewpoints were used to structure the results of an ethnographic study [7].

Highly relevant for this research is contextual inquiry (CI) [2], an ethnographically informed approach that focuses on system development. In CI an analyst with a technical background is in charge of analysing existing work practice. Beyer and Holtzblatt [2] define four principles of contextual inquiry – context, partnership, interpretation and focus – which guide the analyst in interacting with stakeholders and highlight important aspects of this interaction.

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 147 – 161, 2009.

Although contextual inquiry and similar contextual approaches support analysts in understanding the users' work context there still exist several research issues [3]. For instance, researchers highlight that there is no apparent theoretical structure underpinning the observation process. Also, due to a lack of focus these approaches are confined to relatively small-scale environments (e.g. control rooms) [6]. Moreover, ethnographically informed approaches lack on-site software tool support for guiding analysts and for documenting gathered information. Finally, most contextual approaches are weakly integrated with existing requirements engineering (RE) methods and tools, which limits their applicability.

We developed the Mobile Scenario Presenter (MSP) to address these problems. The MSP is an RE tool for mobile devices which supports on-site analysts in contextual requirements elicitation [8]. The evaluation of the MSP tool has included several field- and case studies. For example, at Belfast City Airport the MSP was successfully used to gather requirements for the VANTAGE air traffic management system [8]. Based on lessons learned from these applications of the MSP we developed a contextual method that guides analysts to prepare and conduct scenario-based on-site inquiries. This method was applied to gather requirements for the SemWay ski tour navigation system. We report on this inquiry and compare its results to results from other requirements elicitation activities within SemWay and to results from previous projects. Based on the application of our method and our results we report lessons learned and draw first conclusions on the use of our contextual method in real-world projects.

The paper is structured as follows: Section 2 presents the tool-supported contextual requirements elicitation method ART-SCENE CoRE. Section 3 introduces research questions and reports on the SemWay case study. In Section 4 we analyse the results of this study. Section 5 revisits the research questions. In Section 6 we present lessons learned and conclusions.

2 ART-SCENE CoRE

ART-SCENE CoRE is based on ART-SCENE (Analysing Requirements Trade-offs: Scenario Evaluations) and provides support for Contextual Requirements Elicitation (CoRE). ART-SCENE is a scenario-driven approach to discover requirements and provides software tools for generating and walking through scenarios [9]. The automatically generated scenarios include normal course events (e.g. *The ski hiker starts the navigation of the route*) and alternative course what-if questions (e.g. *What if the ski hiker has some unusual physical characteristics that affect his/her behaviour during this action?*). ART-SCENE focuses on workshop settings: stakeholders, guided by a facilitator and supported by a scribe, walk through scenarios event by event to discover and capture requirements [9]. Mavin and Maiden [10] argue that scenario walkthroughs are effective because people are better at identifying errors of commission rather than omission. ART-SCENE scenario workshops on average generate 10 requirements per hour [10]. However, coming together in a workshop to elicit requirement is not always possible, and is usually costly and time consuming. We developed the MSP, a mobile tool supporting on-site scenario walkthroughs to address these issues. The MSP (see Figure 2) provides the most essential scenario walkthrough capabilities (e.g. view normal/alternative course event, add requirement) to mobile analysts [11]. The lessons learned in previous MSP studies allowed us to

develop ART-SCENE CoRE. This method follows the principles of contextual inquiry [2] and supports analysts in planning and conducting on-site scenario walkthroughs. ART-SCENE CoRE increases guidance for on-site analysts by combining the benefits of scenario-based and contextual requirements elicitation. Moreover, tool support is available in the form of the MSP, but can also be provided by other tools which support ART-SCENE CoRE activities (e.g. navigating scenarios). ART-SCENE CoRE does not intend to replace existing ART-SCENE scenario workshops [9], but complements them by providing scenario walkthrough capabilities for on-site analysts. In particular, the method includes the two contextual activities *on-site scenario validation* and *on-site scenario walkthrough*. These activities cannot be performed in isolation and are based on ART-SCENE activities tailored to the needs of a contextual method (see Figure 1). Each of the four ART-SCENE CoRE activities consists of several steps. We describe these activities using the Entry-Task-Validation-Exit (ETVX) notation [12].

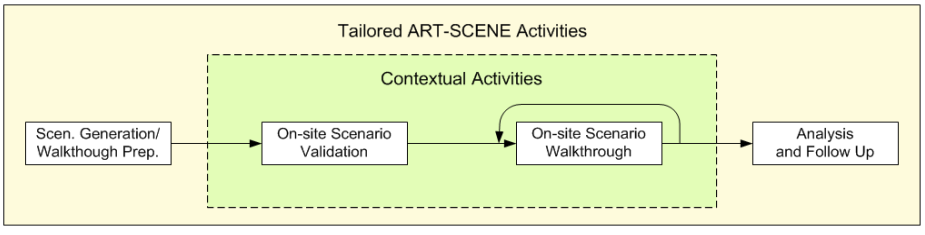


Fig. 1. High level view on ART-SCENE CoRE

Activity 1: Scenario generation and walkthrough preparation

Generate scenarios: ART-SCENE CoRE relies on early ART-SCENE steps such as domain modelling and automatic scenario generation [9]. In particular this first step is not different to other ART-SCENE projects. ART-SCENE's automatic scenario generation capabilities significantly contribute to ART-SCENE CoRE. Scenarios provide a high-level context model supporting on-site inquiries by providing focus for analysts and allowing them to document discovered requirements in a structured way.

Identify scenarios for on-site walkthroughs: After the initial scenario generation analysts identify scenarios suitable for on-site walkthroughs. For instance, scenarios are highly relevant if most of their events correspond to tasks observable on-site. Similar to contextual inquiry we favour so-called normal tasks for on-site inquiries. Normal tasks, such as writing a letter, can be scheduled and interrupted [2]. Most other tasks (e.g. extremely long tasks or uninterruptable tasks) limit the analyst's ability to perform an on-site inquiry.

Train analysts on tools: On-site analysts applying ART-SCENE CoRE must be familiar with the provided tool support. For instance, if ART-SCENE CoRE is supported by the MSP, analysts might need training on this tool before performing an on-site scenario walkthrough.

Selecting interviewees: Analysts need to select stakeholders to involve in the on-site inquiry. Scenario events support this activity as they describe agents (roles)

performing an action [9]. We recommend proceeding as described in [2], where the analyst identifies two or three on-site stakeholders for each role.

Table 1 summarizes the scenario generation and walkthrough preparation activity according to the ETVX concept.

Table 1. ETVX cell for activity 1: Scenario generation and walkthrough preparation

Entry	• Use case specifications • List of potential stakeholders
Task	• The analyst generates scenarios based on use case specifications • The analyst identifies scenarios relevant for on-site walkthroughs • The analyst gets training on provided tool support • Based on the list of potential stakeholders and the selected scenarios an analyst decides on stakeholders to involve in on-site walkthroughs
Ver.	• Normal and alternative course events generated • Scenarios selected for on-site walkthroughs • The analyst is familiar with provided tool support for ART-SCENE CoRE • Stakeholders identified and invited
eXit	• Scenarios including normal and alternative course events • List of stakeholders involved in the on-site walkthroughs

Activity 2: On-site scenario validation

Identification of misleading or missing scenario events: ART-SCENE focuses on developing high quality use case models and scenarios [9]. However, as a first contextual activity we recommend validating the scenarios on-site before starting scenario walkthroughs. Ideally, work practices relevant for system design are reflected by scenario events. Further, the sequence of normal course events should correspond to the observed work tasks. By observing stakeholders' work an analyst checks whether the defined normal course events refer to current work practices. The analyst can also validate alternative course what-if questions to prune irrelevant questions or to identify new questions complementing the automatically generated ones. If analysts identify a gap between ongoing work practices and scenarios they are advised to perform on-site interviews to understand the reasons.

Scenario update: After the on-site scenario validation the analyst performs a scenario update, which includes generating what-if questions for newly discovered normal course events.

Table 2 describes the on-site scenario validation activity.

Table 2. ETVX cell for activity 2: On-site scenario validation

Entry	• Scenarios including normal and alternative course events
Task	• The on-site analyst identifies misleading or missing scenario events • The analyst updates the scenario based on change requests
Ver.	• Change requests documented • Scenarios updated according to change requests
eXit	• Validated scenarios including normal and alternative course events

Activity 3: On-site scenario walkthrough

Conventional interview: Similar to contextual inquiry [2] the on-site analyst first introduces herself to the stakeholders and explains her role. Further, the analyst clarifies the focus of the inquiry by discussing relevant scenario events. This also allows the analyst to present the MSP or other tools to the stakeholders.

Transition: After defining the focus of the inquiry the analyst and the stakeholders discuss how to conduct the actual on-site scenario walkthrough. This includes agreeing on when to interrupt stakeholders' work to discover requirements [2]. Ideally the observed tasks can be interrupted anytime for discussions.

Scenario walkthrough: On-site scenario walkthroughs combine observations and interviews based on scenarios. The analyst observes current work practices and identifies the corresponding normal course events. While observing the ongoing task the analyst takes notes on aspects of work relevant for system design. More importantly, the ongoing tasks are discussed with stakeholders to identify new requirements in due course. This includes asking stakeholders how the future system shall handle normal course events corresponding to their actual work tasks. Further, the analyst discusses alternative course what-if questions, which provide essential input for the discussion of unusual and unexpected system behaviour.

Documentation of requirements and inquiry notes: ART-SCENE CoRE goes beyond mere requirements descriptions [13]. In addition to fully specified text-based requirements descriptions, the analyst documents upcoming requirements with multimedia annotations and text-based information cues. For instance, using text-based information cues significantly reduces the time needed for requirements documentation as analysts only note key information about a requirement and develop a more precisely formulated requirement later.

Table 3 summarizes key aspects of the on-site scenario walkthrough activity.

Table 3. ETVX cell for activity 3: On-site scenario walkthrough

Entry	• Validated scenarios including normal and alternative course events • List of stakeholders involved in the on-site walkthroughs
Task	• The on-site analyst starts the interview and explains the scenario walkthrough to the stakeholders • The on-site analyst performs scenario walkthroughs to discover system requirements and comments from stakeholders • The on-site analyst documents the gathered requirements with the help of multimedia descriptions and information cues
Ver.	• All scenario events discussed in the walkthrough • Upcoming requirements documented
eXit	• System requirements and comments

Activity 4: Analysis and follow up

Transcribe audio recordings and information cues: After the walkthrough the analyst analyses the gathered information. For example, the analyst can use ART-SCENE desktop tools to transcribe audio recording and information cues.

Validate requirements in workshops: Finally, the analyst may validate the gathered requirements in workshops (e.g. ART-SCENE workshops) to discuss them with a larger number of stakeholders.

Table 4 describes the analysis and follow up activity.

Table 4. ETVX cell for activity 4: Analysis and follow up

Entry	•	System requirements and comments
Task	•	The analyst transcribes information cues and audio recordings into precise requirements descriptions
	•	Analysts and stakeholders validate the gathered requirements in workshops and discuss comments
Ver.	•	All requirements transcribed
	•	All requirements discussed in workshops
eXit	•	Precisely defined and validated system requirements

3 The SemWay Project – Discovering Requirements in the Wild

We applied ART-SCENE CoRE within SemWay (Semantic Way), a project in which Salzburg Research, the Technical University of Vienna, and industry partners jointly explore cognitive processes of human wayfinding to develop electronic navigation support for several domains [14]. In particular, the SemWay project sought to develop a ski tour navigation system. To support requirements discovery for this system we applied ART-SCENE CoRE in the Austrian Alps and undertook the ART-SCENE CoRE activities described in Section 2. The only exception was the *on-site scenario validation activity* which was performed in a workshop with domain experts rather than on-site, due to time constraints. In this workshop we adjusted normal and alternative course events according to comments from domain experts. We explored the following research questions in the SemWay on-site scenario walkthroughs:

Q1: Does ART-SCENE CoRE support a single analyst to discover requirements from stakeholders while moving about the domain under analysis and walking through scenarios with the MSP?
Q2: Does ART-SCENE CoRE trigger additional requirements not discovered in previous SemWay requirements elicitation activities?
Q3: Does ART-SCENE CoRE trigger a larger number of requirements per hour of stakeholder participation compared to the average ART-SCENE requirements generation rate?

We chose to investigate Q1 based on issues we identified in earlier evaluation studies. On-site analysts found it difficult to observe and acquire responses from end-users whilst navigating the scenario and entering requirements into the MSP [15]. Therefore in previous VANTAGE and APOSDLE projects we introduced the role of facilitator and scribe for on-site scenario walkthroughs [11]. However, we aimed at providing a method, which would also allow a single analyst to gather requirements in the workplace of future system users with the help of the MSP.

We had already sought answers to Q2 and Q3 in previous research. Results from the VANTAGE and APOSDLE projects allowed us to answer both questions with a tentative yes [11]. However, in VANTAGE and APOSDLE, the on-site inquiry was not based on a clearly defined contextual method such as ART-SCENE CoRE. Instead on-site analysts intuitively used the MSP based on their scenario workshop skills. In SemWay on-site analysts were familiar with ART-SCENE CoRE and followed its predefined steps to interact with stakeholders and to discover requirements.

Fig 2. The MSP displaying SemWay's normal and alternative course events

In contrast to VANTAGE and APOSDLE, where ART-SCENE workshops were held prior to on-site inquiries, Salzburg Research and their partners used several other approaches to gather requirements for SemWay. These included testing and analysis of existing navigation tools, wayfinding experiments and brainstorming workshops [14], which produced 31 requirements (e.g. *The system must detect if a user ski hiker leaves the predefined route*). We agreed with Salzburg Research to complement their RE activities with ART-SCENE CoRE. In order to prevent the on-site inquiry from being influenced, we decided not to access the previously gathered requirements.

Following ART-SCENE CoRE's *scenario generation and walkthrough preparation activity* we identified two scenarios relevant for on-site walkthroughs. *Sc1 Plan route* describes how a skier selects an appropriate route. *Sc2 Navigate route* (see Figure 2) focuses on the skier's navigation tasks while walking uphill on skis.

We conducted an on-site inquiry in the Austrian Alps near Untertauern and decided to form two groups for the on-site inquiry (see Table 5). Each group was headed by a single analyst and included two stakeholders walking up the mountain on skis. Both analysts were trained Personal Digital Assistant (PDA) and MSP tool users. The analysts were equipped with snowshoes and followed the ski hikers, who were also

involved in previous requirements discovery activities for SemWay. Following ART-SCENE CoRE's *on-site scenario walkthrough activity* the analysts first clarified that the focus of the inquiry was the same as in previous SemWay requirements elicitation activities – gathering requirements for the SemWay navigation support system. Moreover the analysts presented the MSP to on-site stakeholders. Both groups agreed that while walking uphill the analyst was allowed to interrupt the ski tour for questions. Analysts observed the movements of the ski hikers in the environment. Based on this context the analysts identified corresponding normal course events and discussed them with the stakeholders. Depending on the steepness, ski hikers either slowed down or stopped for discussions. Requirements were either documented with audio recordings or information cues. Analyst 1 documented requirements using information cues, while Analyst 2 used the MSP's audio recording feature. After the on-site walkthroughs we performed ART-SCENE CoRE's *analysis and follow up activity* to compare and validate the discovered requirements in a workshop.

Table 5. SemWay scenario walkthrough schedule

Date	Scenario	Way of Use
27/02/08	Sc1: Plan route	Work context – Analyst 1
27/02/08	Sc2: Navigate route	Work context – Analyst 1
27/02/08	Sc1: Plan route	Work context – Analyst 2
27/02/08	Sc2: Navigate route	Work context – Analyst 2

4 SemWay Results

Although the inquiry was planned in detail, several unforeseeable challenges occurred that were untypical for requirements discovery and not related to ART-SCENE CoRE. The first challenge was walking with snowshoes. We expected to reach the starting point of the ski tour in less than 2 hours. However, it took the analysts more than 3 hours to get there. The exhausting walk to the starting point of the ski tour reduced time for the on-site walkthroughs, as the analysts needed more time to recover than expected. Considering the short days in winter, instead of almost 3 hours only 1 hour was left for requirements discovery during the ski tour (see Figure 3).

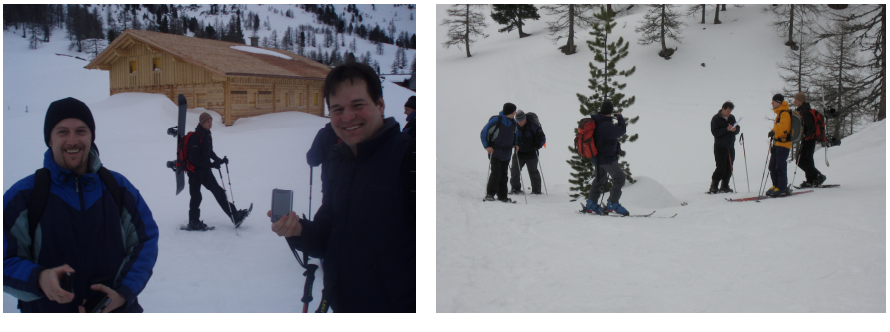


Fig. 3. Requirements discovery in the Austrian Alps

Analyst 1 spent approximately 50 minutes undertaking scenario walkthroughs while Analyst 2 had about 1 hour available to gather requirements on SemWay. The reduced time for Analyst 1 was the result of another problem: after about half an hour the PDAs ran out of battery due to the cold conditions. The analyst then obtained printouts of the scenario events that included alternative course what-if questions from the nearby base camp. So, instead of walking through the scenario events with the MSP, the analysts switched to a paper-and-pencil approach for the rest of the inquiry. Although not planned in advance, this problem allowed us to analyse the suitability of ART-SCENE CoRE with and without tool support.

Despite these problems when performing the on-site scenario walkthroughs in the Austrian Alps, the two analysts discovered 58 requirements on the SemWay navigation support system (see Table 6), including requirements discovered using tool-supported and paper-and-pencil-based on-site inquiries. The group headed by Analyst 1 spent about 25 minutes on each scenario and discovered 17 requirements (7 triggered by what-if questions) for Sc1 and 10 requirements (3 triggered by what-ifs) for Sc2. The second analyst gathered 15 requirements (4 triggered by what-ifs), for Sc1 in 15 minutes and 16 requirements (3 triggered by what-ifs) for Sc2 in 45 minutes.

Table 6. Results of on-site walkthroughs in SemWay

	Total number of requirements documented	Requirements on normal course	Requirements on alternative course	Average number per normal course event
Sc1 – Analyst 1	17	10	7	2.8
Sc2 – Analyst 1	10	7	3	1.6
Sc1 – Analyst 2	15	11	4	2.5
Sc2 – Analyst 2	16	13	3	2.6
Total	**58**	**41**	**17**	

We examined the requirements generated using MSP-supported ART-SCENE CoRE and the paper-and-pencil-based method in turn. Both analysts applied MSP-supported ART-SCENE CoRE when walking through Sc1 and Analyst 2 applied the tool-supported method for 15 minutes when walking though Sc2 to gather 8 requirements (3 triggered by what-ifs). From this data we compute that applying MSP-supported ART-SCENE CoRE on the average generated 21.8 requirements per hour of stakeholder participation. Approximately 35% of these requirements were triggered by what-if questions. Applying the paper-and-pencil-based method had several side effects. Analysts had to remove their gloves to turn pages and write. Moreover it was challenging to read, write and walk uphill at the same time. This slowed down the ski tour and limited the analyst's ability to focus on the stakeholders. Although analysts stopped to document requirements these written information cues were hard to read and it was not always obvious to which scenario event they belonged. Another issue was that only 3 requirements were triggered by what-if questions. Analysts reported that they were not able to browse the printouts of alternative courses and therefore they focused on what-if questions from memory. These problems significantly decreased the average generation rate (9.8 req. per hour of stakeholder involvement).

After the ART-SCENE CoRE on-site walkthroughs, colleagues from Salzburg research made the 31 requirements available, which had been collected in previous SemWay RE activities. These requirements included functional (16 requirements), business (9), usability-type (5), and one safety requirement. Although the focus of the elicitation did not change, ART-SCENE CoRE on-site walkthroughs discovered reliability (4) and interoperability requirements (2). Considering the total number of requirements gathered, more usability-type requirements (14) were found on-site, the number of safety requirements gathered was similar (2), but business goals significantly decreased (1). Again, most of the requirements were functional (35).

For further qualitative analysis regarding requirements' *subjects*, *themes* and *geographical references* we removed all duplicates. There were three duplicate requirements compared to results from previous SemWay RE activities and 7 from the two on-site scenario walkthroughs. This means that in total we found 48 new requirements for SemWay applying ART-SCENE CoRE.

We analysed the *subject* of each requirement and discovered that there were few differences in requirements subject from previous SemWay RE activities and ART-SCENE CoRE on-site inquiries. Most of the requirements found had the SemWay navigation system as their subject (e.g. *The SemWay navigation system shall allow to synchronise routes with other computer systems*). Subjects also included the users and the route (e.g. *The user shall be able to sort routes regarding duration, difficulty and destination*). Only previous SemWay RE activities gathered requirements where the subject was navigation advice, whilst ART-SCENE CoRE generated an additional requirement on GPS and environmental information (e.g. *GPS positioning shall detect the position of the ski hiker in a similar time as car navigation systems*).

We then analysed the requirements' *themes* (see Table 7) and discovered that previous SemWay RE activities gathered more requirements on how the SewWay system navigates its users. However, ART-SCENE CoRE discovered requirements on several other themes. This included 4 requirements on the navigation device and its resistance to outdoor exposure (e.g. *The SemWay navigation device shall be shock resistant*), the navigation corridor (e.g. *The SemWay system shall inform the user when s/he leaves the navigation corridor*) and requirements on environmental impacts on navigation. The on-site walkthroughs also discovered significantly more requirements on information relevant for the tour (e.g. weather forecast and avalanche warnings).

Table 7. Requirements for SemWay by theme

	Previous SemWay RE activities	ART-SCENE CoRE walk-throughs	
Requirements themes	-	Sc1	Sc2
Define route	7	11	0
Navigate tour	19	0	10
Determination of position	2	5	0
Information relevant for the tour	1	5	2
SemWay system	2	0	1
Resistance to outdoor exposure	0	4	0
Navigation corridor	0	0	7
Envtl. impacts on navigation	0	0	3

Of the 31 requirements generated from the previous SemWay RE activities, 8 contained a *geographical reference* (see Table 8). Most referenced the decision point: a point where the skier has to decide which route to take. However, more requirements gathered on-site contained a geographical reference (some of them even had 2 references). In Sc1 we identified 10 requirements out of 25 containing geographical references, four requirements referenced the local area in which the skier was planning to make a tour (e.g. *The SemWay system shall provide a weather forecast for the area of interest*). In Sc2 18 out of 23 requirements contained geographical references, e.g. we found several requirements that addressed the navigation corridor and landmarks (orientation points).

Table 8. Total number of requirements that reference geographical features

	Previous SemWay RE activities	ART-SCENE CoRE walk-throughs	
Geographical reference	-	Sc1	Sc2
Decision point	4	0	2
Landmarks	1	0	6
Start position	0	2	0
Destination	1	2	0
Actual position	2	2	3
Navigation corridor	0	0	7
Local area	0	4	0
Total	**8**	**10**	**18**

5 Research Questions Revisited

Although on-site analysts had to deal with several unforeseen challenges, applying ART-SCENE CoRE helped to discover new requirements for SemWay. Using the results we sought to answer the three research questions.

Can ART-SCENE CoRE be used by a single analyst? In SemWay the two analysts interacted with stakeholders and documented requirements without the support of scribes. The analysts, both experienced PDA users and familiar with the MSP tool were able to focus on the ski tour, communicate with ski hikers, browse the scenarios and to document upcoming requirements, all at the same time. Analyst 2 was even able to walk uphill while documenting requirements with the help of the MSP's audio recording feature. In contrast to previous studies [15], we conclude that the answer to Q1 is yes.

Both on-site analysts agreed that ART-SCENE CoRE provided guidance and support for conducting an on-site scenario walkthrough. In particular analysts highlighted that the short and precise scenarios provided by the scenario validation activity enabled fast navigation and selection of relevant normal and alternative course events. As recommended in the ART-SCENE CoRE method, on-site analysts considered their intensive PDA and MSP training to be an important prerequisite for successful on-site scenario walkthroughs without a scribe.

Does ART-SCENE CoRE trigger additional requirements? Although previous SemWay requirements elicitation activities and ART-SCENE CoRE walkthroughs shared the same focus we were able to discover 48 new requirements on SemWay. We analysed these requirements according to their subject, theme and geographical reference and found several differences. For instance, ART-SCENE CoRE discovered requirements on the navigation corridor and the navigation device' resistance to outdoor exposure, two themes not discussed in previous SemWay requirements. Furthermore requirements gathered with ART-SCENE CoRE contained significantly more geographical references than requirements discovered with previous SemWay requirements elicitation activities. Based on this analysis we conclude that Q2 can be answered with yes – applying ART-SCENE CoRE enabled us to identify new requirements for SemWay.

This result is consistent with the results from previous VANTAGE and APOSDLE studies [11]. However, in previous studies, the gathered requirements were compared to requirements gathered with the help of ART-SCENE scenario walkthroughs, while in SemWay the comparison was based on requirements discovered with different approaches (e.g. wayfinding experiments and brainstorming workshops).

Does ART-SCENE CoRE gather requirements at a higher generation rate? In previous studies [11] we compared the average generation rate of MSP-based on-site inquiries with ART-SCENE scenario workshops. In SemWay no ART-SCENE scenario workshops took place at all. Therefore, we compared the SemWay results with the average ART-SCENE generation rate and to results from previous applications of the MSP.

In SemWay, MSP-supported ART-SCENE CoRE on average generated 21.8 requirements per hour of stakeholder participation, which is significantly higher than the average ART-SCENE generation rate of 10 requirements per hour [10]. Considering that a minimum of 2 stakeholders attend ART-SCENE workshops, the typical generation rate is below 5 requirements per hour of stakeholder involvement. Moreover, SemWay even exceeds the results from previous VANTAGE and APOSDLE studies. Whilst the VANTAGE study produced 8.2 requirements per hour of stakeholder involvement, the two on-site APOSDLE walkthroughs in Graz and Dortmund had generation rates of 12.7 and 17.7 requirements respectively [11]. Based on this result we conclude that Q3 can be answered with a tentative yes – applying ART-SCENE CoRE enables us to gather requirements at a rate higher than the average ART-SCENE generation rate.

Threats to Validity. There are several possible threats to the validity of the reported results. We consider the short duration of SemWay on-site inquiries to be the most critical threat to validity. Instead of having several hours for the on-site inquiry the unexpected long walk to the start point and the problems with PDA runtime significantly limited the time for applying MSP-supported ART-SCENE CoRE. In particular, the short duration of the inquiries might have influenced the calculated requirement's generation rate and therefore our answer to Q3 needs to be interpreted with care. In particular, researchers report that an elicitation process may slow down over time [16]. This means that the short duration of the on-site inquiry could have boosted the generation rate. Another issue is that researchers who have an interest in

the success of ART-SCENE CoRE applied the new method on-site. However, the generation rate was similarly high for both on-site analysts and seems to be in line with results from previous studies. Further research is needed to provide a clearer answer to Q3 as budget, time and stakeholder availability constraints in SemWay did not allow us to undertake further investigations.

In previous studies [11] ART-SCENE workshops were held prior to on-site scenario walkthroughs, and these results allowed us to directly compare ART-SCENE workshops to on-site inquiries. In SemWay, other requirements elicitation techniques than ART-SCENE were applied prior to ART-SCENE CoRE. We are unaware of details of these requirement elicitation activities and how they might have influenced the quality of the gathered requirements and therefore the qualitative analysis presented in this paper. Another issue regarding the qualitative analysis is that about 30% of the considered requirements came from paper-and-pencil-based scenario walkthroughs. We decided to include these requirements in the analysis because analysts still followed a variant of ART-SCENE CoRE, which produced requirements equivalent in their types, subjects and themes.

6 Lessons Learned and Conclusion

The conducted research confirms results from previous studies [11]. Moreover we are able to present lessons learned that highlight benefits and weaknesses of applying ART-SCENE CoRE to gather requirements for SemWay.

Analysts need to be familiar with the environment. The SemWay project highlighted the importance of being familiar with the environment where the on-site scenario walkthrough takes place. In SemWay it could have helped us to identify issues such as the reported battery problems and the long walk to the starting point. ART-SCENE CoRE's on-site scenario validation supports the investigation of the environment as analysts validate scenario events in the workplace of future system users. However, in SemWay we replaced ART-SCENE CoRE's on-site scenario validation with a scenario validation workshop due to time constraints. We now conclude that SemWay's validation workshop was not an effective substitute for on-site scenario validation, as analysts were not able to get sufficiently familiar with the environment.

Mobile tools better support on-site scenario walkthroughs. Due to battery problems caused by the cold conditions, the MSP-supported ART-SCENE CoRE scenario walkthroughs had to be replaced with a paper-and-pencil-based method. Therefore the analysts used scenario printouts, which were also used to document upcoming requirements. This caused several problems as reported above and might be the main reason for the significantly decreased average generation rate. We therefore conclude that tool support on a small and mobile device provided better support for mobile analysts and increased the efficiency of on-site scenario walkthroughs.

ART-SCENE CoRE requirements cover unexpected system behaviour. In SemWay, alternative courses provided essential input for requirements discovery as about 35% of the generated requirements were triggered by what-if questions. This is a rate significantly higher than in previous on-site inquiries (e.g. 19.7 and 17.1 in APOSDLE).

We conclude that in SemWay, ART-SCENE CoRE supported analysts in discovering requirements covering unexpected and unusual system behaviour.

Reasons for the high number of requirements generated by alternative course events include that scenario validation workshop supported the creation of short and focused scenarios. In particular we deleted automatically generated what-if questions considered to be irrelevant and added new ones specific to the domain (e.g. *What-if the weather changes?*). Moreover, domain experts highlighted the importance of particular what-if questions, encouraging analysts to focus on these questions during the on-site inquiry. Analysts reported that the short and focused scenarios also eased navigation between normal and alternative course events.

Audio recording speeds up on-site scenario walkthroughs. As reported, one analyst used audio recordings to document upcoming requirements while the other analyst used text based recognition cues. The requirements generation rate of the analyst using audio recording was significantly higher. Apart from that, using audio notes also enabled the analyst to keep walking uphill while documenting requirements. This increased the speed of the ski tour itself and the group around this analyst was much faster. We recommend the intensive use of audio recordings to document upcoming requirements as in SemWay it increased the requirements generation rate and allowed the analyst to document requirements while moving around.

In this paper we presented the tool-supported ART-SCENE CoRE method that supports mobile analysts to perform structured on-site scenario walkthroughs. The contextual method was used to gather requirements for the SemWay ski tour navigation system. We analysed the results of this inquiry in order to answer the research questions raised. Moreover, we presented lessons learned based on our experiences during the on-site scenario walkthroughs.

ART-SCENE CoRE was developed based on lessons learned from earlier MSP-supported on-site inquiries where analysts followed an intuitive approach to discover requirements. There are obviously overlaps between previous studies and the application of ART-SCENE CoRE in SemWay. Nevertheless, with ART-SCENE CoRE we developed a structured contextual requirements elicitation method supported by a mobile software tool. With the help of this work we have shown that ART-SCENE CoRE is able to support analysts in performing on-site inquiries without the help of a scribe. Moreover, it supported us to find additional requirements for the SemWay navigation support system. However, future research and applications of ART-SCENE CoRE will be needed to explore the method in more detail. This includes investigating Q3 in more detail, as well as research on the repeatability of the ART-SCENE CoRE method, and more importantly the relative benefits and costs of one analyst versus two analysts using the MSP tool during on-site scenario workshops. We look forward to reporting these results in the future.

References

1. Agar, M.: The Professional Stranger: An Informal Introduction to Ethnography. Academic Press, New York (1980)
2. Beyer, H., Holtzblatt, K.: Contextual Design: Defining Consumer-Centered Systems. Morgan Kaufmann, San Francisco (1998)

3. Viller, S., Sommerville, I.: Social Analysis in the Requirements Engineering Process: From Ethnography to Method. In: 4th IEEE Symposium on Requirements Engineering, pp. 6–13. IEEE Computer Society, Washington (1999)
4. Crabtree, A.: Ethnography in Participatory Design. In: 5th Biennial Participatory Design Conference (PDC), CPSR, Seattle, Washington, USA, pp. 93–105 (1998)
5. Bentley, R., Hughes, J., Randall, D., Rodden, T., Sawyer, P., Shapiro, D., Sommerville, I.: Ethnographically-Informed Systems Design for Air Traffic Control. In: 5th Conference on Computer Supported Cooperative Work (CSCW), pp. 123–129. ACM, New York (1992)
6. Heath, C., Luff, P.: Collaboration and Control: Crisis Management and Multimedia Technology in London Underground Control Rooms. In: 5th Conference on Computer Supported Cooperative Work (CSCW), pp. 69–94. ACM, New York (1992)
7. Hughes, J., King, V., Rodden, T., Anderson, H.: The Role of Ethnography in Interactive Systems Design. In: ACM Interactions, vol. 2(2), pp. 56–65. ACM, New York (1995)
8. Maiden, N., Ncube, C., Kamali, S., Seyff, N., Grünbacher, P.: Exploring Scenario Forms and Ways of Use to Discover Requirements on Airports that Minimize Environmental Impact. In: 15th IEEE International Requirements Engineering Conference, pp. 29–38. IEEE Computer Society, Washington (2007)
9. Maiden, N.: Systematic Scenario Walkthroughs with ART-SCENE. In: Alexander, I., Maiden, N. (eds.) Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle, pp. 161–178. John Wiley & Sons, Hoboken (2004)
10. Mavin, A., Maiden, N.: Determining Socio-Technical System Requirements: Experiences with Generating and Walking Through Scenarios. In: 11th IEEE International Requirements Engineering Conference, pp. 213–222. IEEE Computer Society, Washington (2003)
11. Seyff, N., Maiden, N., Ncube, C., Karlsen, K., Lockerbie, J., Grünbacher, P., Graf, F.: Exploring How to Use Scenarios to Discover Requirements. Requirements Engineering Journal (REJ) (to be published)
12. Radice, R.A., Roth, N.K., O'Hara Jr., A.C., Ciarfella, W.A.: A Programming Process Architecture. IBM Systems Journal 24, 79–90 (1985)
13. Rabiser, R., Seyff, N., Grünbacher, P., Maiden, N.: Capturing Multimedia Requirements Descriptions with Mobile RE Tools. In: 1st International Workshop on Multimedia Requirements Engineering - Beyond Mere Descriptions (MeRE), pp. 2–7. IEEE Computer Society, Washington (2006)
14. Rehrl, K., Leitinger, S., Gartner, G.: The SemWay Project - Towards Semantic Navigation Systems. In: 4th International Symposium on LBS & TeleCartography, Hong Kong (2007)
15. Maiden, N., Seyff, N., Grünbacher, P., Otojare, O., Mitteregger, K.: MakingMobile Requirements Engineering Tools Usable and Useful. In: 14th International Requirements Engineering Conference, pp. 26–35. IEEE Computer Society, Washington (2006)
16. Sindre, G., Krogstie, J.: Process heuristics to achieve requirements specification of feasible quality. In: 2nd International Workshop on Requirements Engineering – Foundation of Software Quality (REFSQ), Augustinus, Aachen, pp. 92–103 (1995)

Inventing Requirements with Creativity Support Tools

Inger Kristine Karlsen[1], Neil Maiden[1], and Andruid Kerne[2]

[1] Centre for Human-Computer Interaction Design and Centre for Creativity in Professional Practice, City University, London, UK
kristine.karlsen@soi.city.ac.uk, n.a.m.maiden@city.ac.uk,
[2] Interface Ecology Lab, Texas A&M University, USA
andruid@cs.tamu.edu

Abstract. [**Context and motivation**] Creativity is indispensable for software systems to deliver progress and competitive advantage for stakeholders. Yet it is rarely supported in requirements processes. [**Question/problem**] This paper investigated integration of two software tools, one for generating requirements with scenarios, the other for supporting people to think creatively while finding and collecting information. The effectiveness of the integration was investigated. [**Principal ideas/results**] The technical integration is described, and an evaluation is reported. [**Contribution**] Results reveal some effect on the novelty of the requirements generated, and have implications for the design of tools to support creative requirements processes.

Keywords: Requiremensts discovery, creativity, creativity support tools.

1 Creating Requirements

Requirements engineering is a creative process in which stakeholders and analysts work together to create ideas for new software systems, which are expressed as requirements. Creativity is indispensable if software systems are to deliver progress and competitive advantage, yet it is rarely supported in requirements tools. In this paper we describe the integration of a requirements tool and a creativity support tool, then report how analysts used the integrated tools to specify requirements of a new secure access system.

Most current requirements processes and tools support problem analysis and system specification [17]. An assumption is that stakeholders already know their requirements. However, this is not always true, because stakeholders are not aware of what new technologies can do. As technologies evolve, stakeholders need to create novel requirements by connecting knowledge of the problem with information about relevant technologies. This means engaging in a form of creativity known as information discovery, in which processes of finding, collecting, and arranging information stimulate the emergence of new ideas [8]. In software development new ideas are often expressed as requirements.

Previously we applied theories and models of creativity in workshops to support stakeholders to discover requirements for complex systems in domains including air

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 162–174, 2009.

traffic management and food traceability. However, although successful in terms of the number and the impact of requirements generated [11, 12], the workshops were resource-intensive. Each could involve up to 20 stakeholders and analysts for 2 days, and our support for creative thinking was not available during other requirements processes.

To support people's creative thinking effectively throughout a requirements process, we sought to deliver new tools. Whilst creativity support tools are available, none have been explicitly built to support requirements processes. Therefore we integrated ART-SCENE, a tool designed to discover more complete requirements with scenarios, with combinFormation, a tool that supports people in creating new ideas while finding and collecting information. The remainder of this paper is in 6 sections. Sections 2 and 3 describe ART-SCENE and combinFormation, and their integration, then section 4 describes how an analyst might use the integrated tools. Section 5 reports results for a preliminary evaluation of the integrated tools. The paper ends with a review of related work and future research plans.

2 Enhancing Scenario Walkthroughs

Scenarios are an effective requirements technique, and ART-SCENE is an internet-based environment that uses scenarios to discover more complete requirements. Stakeholders have applied ART-SCENE successfully to discover requirements on software systems in domains ranging from air traffic control to work-based learning [14].

ART-SCENE delivers two important capabilities to stakeholders. The first is automatic scenario generation. ART-SCENE automatically generates one or more scenarios with different normal course event orderings and alternative courses from a use case specification. The second capability is guided walkthroughs of these generated scenarios. The big idea behind walkthroughs is very simple – that people are better at recognition than recall [1]. ART-SCENE scenario walkthroughs offer stakeholders recognition cues in the form of generated alternative courses. If the alternative course is relevant to the system being specified but not yet handled in the specification, then a potential omission has been identified. ART-SCENE guides the stakeholders to specify more complete requirements.

Automatically generated scenarios in ART-SCENE are delivered to stakeholders using a web client shown in Figure 1. The left-side menu provides functions for viewing the scenario and requirements generated for it. Top-line buttons offer walkthrough functions (e.g. next or previous event) and functions to add, edit or delete events, comments and requirements. The left-hand main section contains the sequence of events of a scenario that describe the behaviour of a system, in this case security access to a building. Event 1 describes the start of an action: *user walks up to the security gate*. The right-hand main section describes generated alternative courses for each normal course event, presented in the form of 'what-if' questions. The top-listed alternative course is *what if the user is physically unable to undertake this action*? Alternative courses are generated for different normal course events. If no requirements are specified to handle an event that is recognised as relevant, then omissions have been discovered and new requirements can be written, thus increasing requirements completeness.

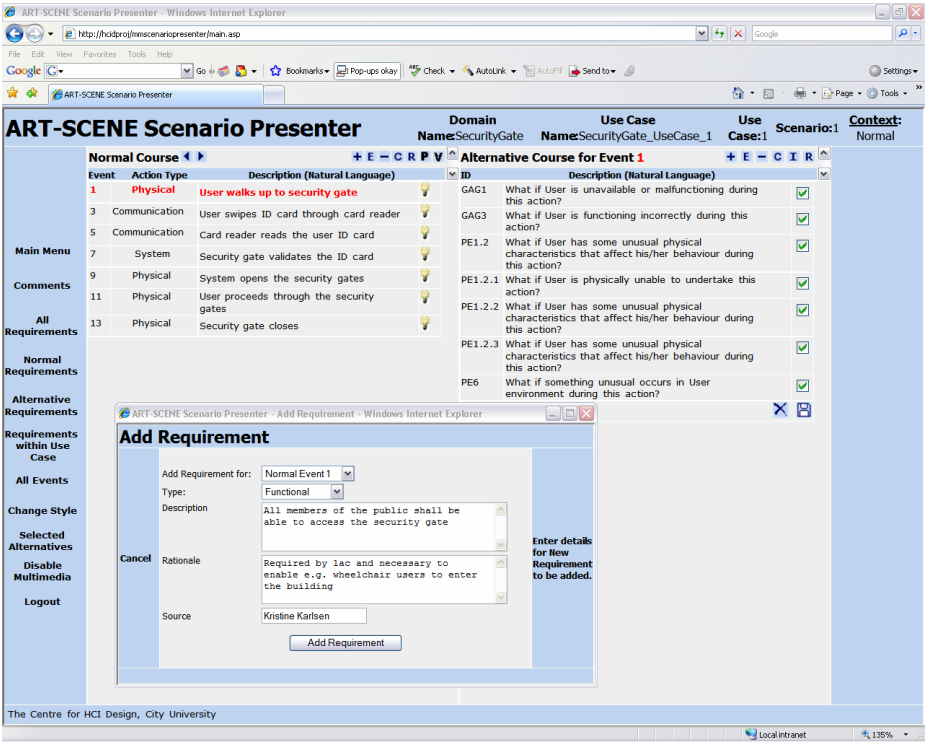


Fig. 1. A snapshot of ART-SCENE, showing one scenario for a security access system, describing the event *user walks up to security gate* (left side), automatically generated alternative courses for the highlighted normal course event (right side), and one generated requirement in a VOLERE form

Returning to the example, if *the user walks up to the security gate*, and *the user is physically unable to undertake this action*, stakeholders might generate new requirements to handle this event, such as *all members of the public shall be able to access the security gate*.

One benefit of using scenarios in ART-SCENE has been improved communication between stakeholders, especially in distributed settings where communication is asynchronous. Scenarios provide a shared context and lingua franca for people in different places to communicate about requirements. To facilitate this communication, ART-SCENE supports a server-side scenarios and requirements database that can be accessed by stakeholders in distributed settings.

3 Extending ART-SCENE with Creativity Support

ART-SCENE did not explicitly support stakeholders in thinking creatively about requirements. Therefore we worked to extend scenario walkthroughs with creativity stimuli, as stakeholders work to develop requirements. Sternberg's defines creativity

as "*the ability to produce work that is both novel (i.e. original, unexpected) and appropriate (i.e. useful, adaptive concerning task constraints)*" [22]. The goal was for digital information stimuli to support stakeholders in creating requirements that are novel and useful.

We integrated ART-SCENE and combinFormation, a creativity support tool for searching, browsing, collecting, mixing, organizing and thinking creatively about digital information [7, 8, 9, 10]. combinFormation is regularly used by 1000 undergraduate design students each year to collect existing work as they create new designs.

3.1 combinFormation

combinFormation is a freely available [7] mixed-initiative [6] system that integrates searching, browsing, and exploring information [9, 10]. It has been developed as an extensible platform with a modular object-oriented architecture. Software agents procedurally extract clippings from documents, which function as surrogates, and assemble them in a visual c*omposition space* storyboard, shown in Figure 2. A *surrogate* is an enhanced bookmark, which represents an important idea in a document, and enables navigation back to the document. Visual surrogates are formed by extracting images, and augmenting them with metadata. *Composition* functions a means for representing a collection of surrogates as a connected whole, instead of as separate elements, as in a list. The visual composition is procedurally generated over time, like a dynamic video. Related surrogates are automatically clustered. Procedural generation iteratively places visual surrogates into the composition space, where the participant can see and manipulate relationships among them. By making ideas and relationships visible, the composition space can stimulate cognitive restructuring, and creative ideation. Design tools enable authoring task-oriented collections as navigable compositions. Visual characteristics, such as colors, sizes, fonts, sizes, layout, and compositing can be adjusted. Spatial relationships and visual characteristics are used to connect ideas. Compositing blends surrogates visually, by using alpha masks.

The user engages in processes of searching, browsing, collecting, and authoring media in the composition space, which serves as a visible medium for communication between human and agent, as well as for thinking about and sharing information resources. Users can directly experience the juxtaposed surrogate clippings, and they can also navigate back to source documents for more in-depth information. Compositions can be saved reopened, shared, and published.

A combinFormation session is initiated by the user through the specification of seeds. Each seed is a search query, website, or news feed. The top left of Figure 2 shows a process of seeding combinFormation, in this case to develop a composition relevant to requirements generation for a secure access system, and the resulting composition space. Queries included terms such as *physical security* and *security gate*. The outer area is the mixed-initiative Hot Space, which is shared by combinFormation's visual composition agent and the user. Surrogates stream directly into this space. The inner area is the Cool Space, available only to the user for constructing a compositional storyboard of the surrogates most relevant to the task at hand, using drag and drop. As the user fills the Cool Space storyboard with relevant content, s/he can enlarge it to allocate more of the visual area for her own use, leaving less for the agent.

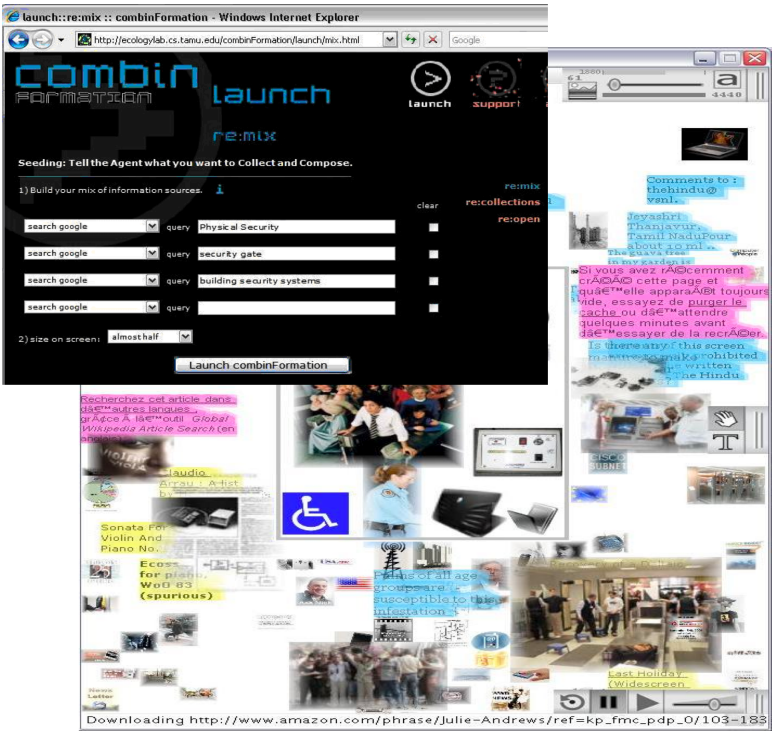


Fig. 2. Launching combinFormation by issuing multiple search queries as seeds for mixing (e.g. "Physical Security" and "security gate") in the top left-hand corner, and the composition space filled with retrieved surrogates. The user organizes relevant surrogates by dragging them into the center Cool Space, and adjusting their visual characteristics and relationships.

As part of integrating ART-SCENE and combinFormation, we developed a new ART-SCENE component called the Creative Requirements Innovation Space (CRIS). CRIS provides analysts with a flexible storyboarding space in which to explore, combine and transform ideas prior to generating requirements.

3.2 CRIS: The Creative Requirements Innovation Space

We designed CRIS to emulate how people in our creativity workshops generate new associations between concepts using creativity techniques. In workshops, stakeholders browse and associate ideas, place them on pin boards, then elaborate these ideas in storyboards that combine graphics and text to describe new system behaviour [12]. Using CRIS the analyst explores associations between image and text elements presented by combinFormation, then pulls these elements into CRIS to connect them with requirements.

So how did the tool integration work? We designed ART-SCENE to extract key terms dynamically from scenario event descriptions, and pass them to combinFormation to seed searches and download web pages to create collections of surrogates and links displayed in its composition space. An analyst then used combinFormation to

manipulate surrogates in the composition space and the visual representation of the collection. To document requirements, the analyst opened another storyboard in CRIS, dragged selected surrogates from the composition space to construct a storyboard for each scenario normal course event, then documented one or more structured requirements through VOLERE forms. The storyboard and requirements were stored in the ART-SCENE database, which met one important need – that storyboards and requirements can be shared between stakeholders working in distributed settings with the same ART-SCENE scenario.

To facilitate rapid storyboarding, the analyst could directly drag elements from the Composition Space and/or any existing web-site into the CRIS storyboard, then resize, label and delete elements, combine elements together, save the storyboard, reopen it later, and share it with other stakeholders. To minimize resource consumption and quicken response times, most CRIS functions were client-side, and client-server communication only happened when the analyst stored a storyboard in the database.

The next section demonstrates how this architecture supports an analyst to generate and use CRIS storyboards during an ART-SCENE scenario walkthrough.

4 ART-SCENE and combinFormation to Invent Requirements

The security access scenario in Figure 1 has 7 discrete normal course events, from *user walking up to a security gate* to *the security gate closes*, and alternative courses generated to ensure requirements completeness. For any selected use case event, the analyst can request creativity support by clicking the corresponding light bulb icon, then edit and extend the terms in the event description – for example *user, walk and security gate*. ART-SCENE seeds combinFormation with the extracted event terms.

The composition space in Figure 2 connects surrogates that include images of a crowd scene, a security camera and a disabled access symbol. The analyst uses combinFormation to organize surrogates in the composition space. She drags some surrogates into CRIS to develop new requirements for security access. Figure 3 shows a CRIS storyboard in ART-SCENE, created from surrogates retrieved and associated from the composition space of Figure 3.

More than one user can walk up to the security gates. The user can interact with the gate in different ways. Disabled access is provided, there are human security guards to provide assistance, and surveillance is in place, providing images of the faces of the registered user of an ID card.

During storyboarding the analyst can formalize new requirements in the VOLERE form. Figure 3 shows two functional requirements, one that specifies that the security guard shall view a picture of the registered user when an ID is swiped (based on images of banks of screens and images of people's faces), whilst the second specifies that the user will access to gate without actively having to swipe or use the card (based on images of a Smart card and fingerprint system). ART-SCENE supports further communication and traceability of requirements by linking each documented requirement and CRIS storyboard in the server-side database.

However we lacked empirical evidence that the integration could deliver requirements that, following Sternberg's definition, are more novel but still useful. Therefore a first exploratory evaluation was undertaken.

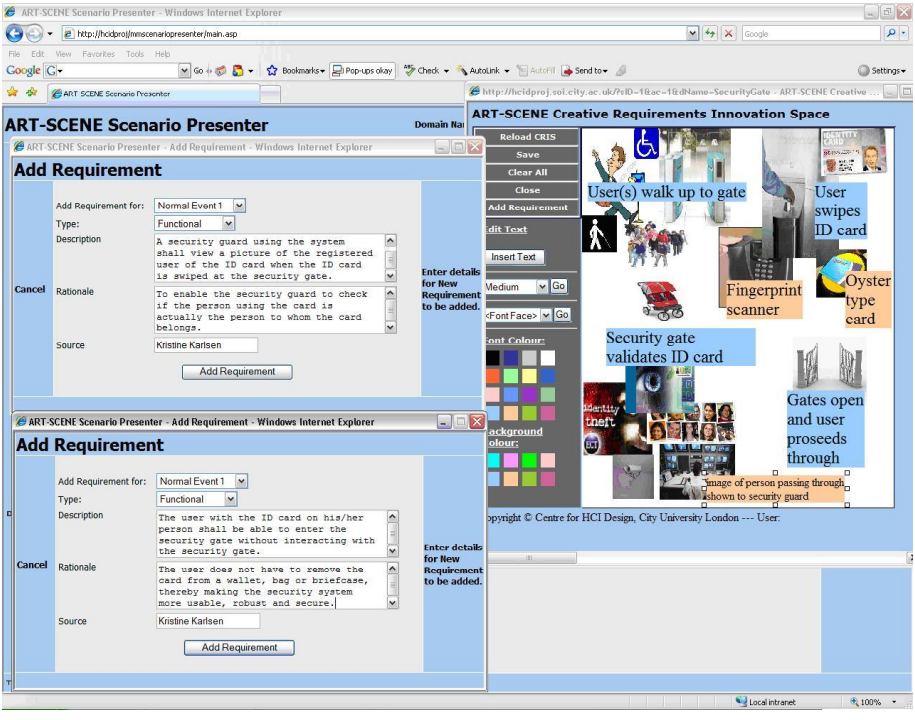


Fig. 3. A storyboard in CRIS, showing elements discovered with combinFormation, annotated with text (in blue) written within CRIS by an analyst. Two possible requirements generated from the CRIS storyboard are specified using VOLERE forms. One specifies the security guard shall view a picture of the registered user when an ID is swiped, whilst the second specifies that the user will be able to access to gate without actively having to swipe or use the card.

5 A First Exploratory Evaluation

Nine analysts worked individually with an environment comprising the ART-SCENE with CRIS and combinFormation (2006 version) tools (referred to ART-SCENE/cF) to generate requirements for the security access system scenario in Figure 1. Each had received training in walking through scenarios and writing VOLERE requirements. Each received a basic introduction to ART-SCENE, combinFormation and CRIS functions.

Requirements generated by the analysts were analyzed to explore 3 research questions based on Sternberg's definition of creativity:

Q1: Do analysts generate more requirements using ART-SCENE/cF than when using ART-SCENE-only?

Q2: Do analysts generate requirements that are more novel using ART-SCENE/cF than when using ART-SCENE-only?

Q3: Do analysts generate requirements using ART-SCENE/cF that would not have been generated using ART-SCENE-only?

The evaluation was in two parts of 20 minutes each. In the first, each analyst walked through the scenario with ART-SCENE-only to discover and document requirements for the security access system. Next, the experimenter seeded combinFormation with queries. The analyst continued the walkthrough in ART-SCENE with combinFormation and CRIS to discover and document requirements. Each analyst was then debriefed. Afterwards, a security expert rated the usefulness and novelty of all requirements generated by the 9 analysts.

All 9 analysts completed both parts of the evaluation and generated at least one CRIS storyboard each. combinFormation queries were simple and not tuned. During debriefings, all 9 analysts claimed that using ART-SCENE/cF had helped them to generate requirements not generated using ART-SCENE-only based on prompts from the composition space. Requirements totals by analyst are reported in Table 1.

Table 1. Totals of requirements generated by each analyst with ART-SCENE-only and with ART-SCENE/cF, and totals of security expert ratings of the usefulness and novelty of these requirements

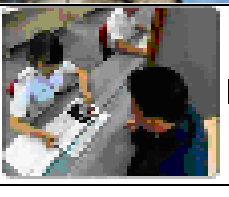
Analyst ID	ART-SCENE-only			ART-SCENE/cF		
	# of Requirements	Usefulness	Novelty	# of Requirements	Usefulness	Novelty
A1	16 (62%)	16 (100%)	0 (0%)	10 (38%)	10 (100%)	0 (0%)
A2	6 (67%)	5 (83%)	1 (17%)	3 (33%)	2 (67%)	0 (0%)
A3	13 (76%)	13 (100%)	0 (0%)	4 (24%)	4 (100%)	0 (0%)
A4	8 (62%)	8 (100%)	0 (0%)	5 (38%)	5 (100%)	0 (0%)
A5	14 (64%)	14 (100%)	0 (0%)	8 (36%)	8 (100%)	0 (0%)
A6	17 (63%)	17 (100%)	0 (0%)	10 (37%)	10 (100%)	0 (0%)
A7	10 (63%)	10 (100%)	3 (30%)	6 (38%)	6 (100%)	1 (17%)
A8	14 (61%)	14 (100%)	0 (0%)	9 (39%)	7 (78%)	3 (33%)
A9	7 (50%)	6 (85%)	1 (15%)	7 (50%)	6 (85%)	2 (29%)
Totals	**105 (63%)**	**103 (98%)**	**5 (5%)**	**62 (38%)**	**58 (94%)**	**6 (10%)**

On average, each analyst generated almost twice as many requirements with ART-SCENE-only as with ART-SCENE/cF. Reasons reported included generating the obvious requirements quickly with ART-SCENE-only, and the longer time needed to learn to use CRIS and select elements from combinFormation.

The security expert rated 98% of the ART-SCENE-only requirements as useful and 5% as novel, and 94% of the ART-SCENE/cF requirements as useful and 10% as novel, so although analysts generated fewer requirements when using ART-SCENE/cF, a higher percentage of these were rated as novel. Three of the 11 novel requirements were also rated as not useful for reasons that included *too much administration* and *not practicable due to costs*.

The 11 novel requirements – 5 generated with ART-SCENE-only and 6 with ART-SCENE/cF – were investigated. We attributed 4 of the 5 ART-SCENE-only requirements to alternative course events generated automatically in ART-SCENE. For example, the requirement *the system shall notify the user and alert an administrator when there is a malfunction with the card reader* in a response to the alternative course event *what if the Card Reader is unavailable or malfunctioning during this action?* One analyst (A7) specified 3 of these 5 novel requirements, suggesting that he exploited alternative course prompts for creative thinking more than the other analysts did.

Table 2. Eight requirements and their rationale generated by analysts using ART-SCENE/cF, and storyboard elements associated with these requirements. The security expert ranked all 8 as useful, and the first 5 as innovative.

Analyst A8 **Description**: If a toll is required, security should accept cash.		**Analyst A8** **Description**: It should be possible for the gate system to know who to expect at a certain time. The face of the person should be displayed on the monitor of the security desk when someone is expected.	
Analyst A8 **Description**: If user looks drastically different to when photographic ID was taken, security should be able to take new photos there and then. **Rationale**:		**Analyst A9** **Description**: When the user approaches the security gate, information will be displayed to them informing them on how to use the system.	
Analyst A9 **Description**: The security gates should have numerous entry points maximising the amount of people that can pass through at once.		**Analyst A9** **Description**: The system will include some CCTV aspect which will record who goes through at what time. it will be time linked with the recorded info in the card reader.	
Analyst A4 **Description**: Human assistance should be available in the event of a technical failure.		**Analyst A1** **Description:** The security gate is connected to a camera which can recognize and match features of a user to their ID card image/profile.	

During debriefings, analysts attributed only 1 of 6 ART-SCENE/cF requirements to alternative course events (analyst A7 again). The remaining 5 were attributed to surrogates in the analysts' storyboards shown in Table 2.

Results revealed individual differences between analysts. Analysts A8 and A9 generated the 5 innovative requirements associated with storyboard elements. For example, the requirement *if a toll is required, the security should accept cash*, was associated with an image of an *briefcase of cash*, whilst the requirement to *display the face of expected visitors* was associated with a *person's face on the retrieved magazine cover*. Similar associations were detected for the other 3 novel requirements, and with requirements rated as useful but not novel (Table 2).

All 9 analysts reported preferences for images over text, citing reasons like *images gave more impact* and *I could look at more pictures at the same time*.

The results provide tentative answers to our research questions. The answer to **Q1** was no, at least in a restricted time period. Using ART-SCENE/cF took more time than ART-SCENE-only, reducing the frequency and number of requirements that analysts generated. There is weak evidence to answer yes to **Q2**, because double the proportion of requirements generated with ART-SCENE/cF were rated as novel. However, introducing ART-SCENE/cF did not result in a step change in requirements creativity. Individual differences between analysts were a factor – all requirements rated as novel were generated by just one-third of the analysts. There is stronger evidence for answering yes to **Q3**. All analysts reported that ART-SCENE/cF prompted the generation of new requirements, both novel and otherwise, in connection with composition space surrogates.

Clearly there are threats to the validity of this first exploratory evaluation. Many of the threats limit the generality of the conclusions that can be drawn. A small number of analysts undertook the evaluation, which restricted our conclusions about the effectiveness of ART-SCENE/cF. The analysts had previous analysis experience within a given range, and our results cannot be applied to inexperienced analysts and analysts with over 10 years of analytic experience. Furthermore the results revealed important individual differences, with most creativity results generated by a small subset of the analysts. Finally the evaluation was undertaken in one problem domain – *secure gate access* – with small numbers of cF queries, and it is difficult to generalize the results to other domains, because the nature of the requirements discovered might be different and because different types of storyboard elements might be retrieved by cF.

Other threats to the validity of the evaluation were due to the design of the evaluation. The number of analysts available meant that a control group was not practical. We were unable to investigate the effect of the order of the two study tasks on requirements discovery, and increased exposure to the scenarios over time might have led to more discovered requirements. Likewise more training in combinFormation and CRIS might increase the volume and novelty of the requirements. In particular, if participants had known about how to express interest in combinFormation, this would have helped them to receive information more relevant to their tasks. combinFormation's fluid interface minimizes the cognitive effort required [8]. Likewise, knowledge of how to automatically generate a new search, using an existing surrogate in combinFormation, would have helped the analysts. Improving integration of the technologies is expected to improve the experience and results.

Nonetheless, the results reveal that the use of ART-SCENE/cF prompted analysts to generate requirements that otherwise might not have been generated. Evidence that only a minority of the ART-SCENE/cF requirements were novel is consistent with findings from earlier creativity workshops [12], in which stakeholders tend to generate useful requirements that are both novel and otherwise during periods of creative thinking.

6 Related Work

Little requirements research has addressed creative thinking directly. Brainstorming techniques and RAD/JAD workshops [3] make tangential reference to creative thinking. Most current brainstorming work refers back to Osborn's text [19] on principles

and procedures of creative problem solving (CPS). Examples of CPS activities include *the matrix*, which involves making lists then selecting items from each list at random and combining them to generate new ideas, and *parallel worlds*, which uses analogical reasoning to generate new ideas. However, there are no reported applications of the CPS model to requirements processes.

In the requirements domain, Robertson [20] argues that requirements analysts need to be inventors to bring about the innovative change in a product that gives competitive advantage. Such requirements are often not properties that a stakeholder would ask for directly. Nguyen et al. [18] observed that teams restructured requirements models at critical points when they re-conceptualize and solve sub-problems, triggered by moments of sudden insight. Mich et al. [15] report the successful use of the elementary pragmatic model from communication theory in a controlled environment to trigger combinatorial creativity during requirements acquisition. The RESCUE requirements process has ran numerous creativity workshops in domains from air traffic management to food information traceability [11, 12].

In creativity research, creativity support tools have been posited to support users to discover, explore, innovate and imagine. Schneiderman [21] reports that use of these tools to locate, study, review and revise can accelerate users' creative efforts. Search engines such as Google are important in such tools to locate information quickly. However Kules [16] reports that these search engines are more effective for retrieving information from well-defined queries than to support creative tasks with incomplete or ambiguous queries in often-unfamiliar domains. That said, by using more than ranked lists of search results, such as the hot space in combinFormation, we can support creativity by exposing users to information that will help the creative process. Greene [4] reported other important characteristics of creativity support tools, which include pain-free exploration and experimentation, supporting engagement with content to promote active learning, iteration, and collaboration. Future requirements tools will need some of these characteristics.

Whilst our work has sought to support individual creativity, much research focuses on collaborative creativity. Mamykina et al. [13] report research results that reveal the importance of social interactions, mentoring and collaboration in creative work, and Fischer & Giaccardi [2] talk of the need to sustain social creativity. Mamykina et al. [13] further report that effective collaboration involved 3 main activities – creative conceptualization, realization and evaluation. Future creativity tools that will support requirements processes might be expected to support at least these activities.

7 Creativity Support Tools for Requirements

We successfully integrated 2 tools to support people to think creatively about requirements. These tools utilize the Internet as a source of descriptions of domain events and situations that stimulate analysts and stakeholders to generate requirements that are more complete and novel.

The evaluation results have implications for improving the design of creativity support tools for requirements analysts. We need much better integration of creativity support tools with the tasks they intended to assist. Analysts' loss of efficiency while generating requirements is a concern. We can improve the efficiency of the

composition's stimulation of creative thinking in requirements generation. We hypothesize that using combinFormation's cool space as the storyboard, and then integrating combinFormation directly with server-side ART-SCENE will improve the experience and results in several ways. First of all, the number of windows will be reduced, thus increasing efficiency by reducing cognitive effort [5]. There is also an opportunity to invoke peripheral attention that minimizes the physical effort to see and interact with the composition being generated, the tools create the opportunity for the analyst to be stimulated by relevant information when it arises. The benefits of mixed-initiative composition on requirements generation will be increased. We will give users fluid mechanisms, next, for associating a combinFormation surrogate with an event in an ART-SCENE scenario, and with a requirement that the user has generated. We will also enable stakeholders working on the same scenario to share compositions.

We expected the analysts to generate one storyboard per scenario event of interest, but this did not happen. Although a lack of time to produce more than one storyboard was a factor, each storyboard included elements related to more than one scenario event. One possible reason was that elements retrieved from the Internet were more coarse grain than the actions, agents and objects described in the scenario events in ART-SCENE. One implication is to use composition to support the earlier use case authoring tasks, using surrogates to provide scaffolds with which to write the use cases.

The most important step is to derive queries automatically from ART-SCENE more effectively and pass them as seeds to the mixed-initiative composition space. The granularity of the queries can start with the generality of a scenario, and the shift to specific events and requirements. For software to perform this shifting automatically, it must model the user's attention in the context of using ART-SCENE to perform a task. This integration will be facilitated by combinFormation's services mechanism, through which the software functions as a composition visualization server that can respond to semantic messages. Thus, ART-SCENE will periodically send XML messages to the running cF over a network socket. The experimental results reveal that effective query formulation is essential to providing elements relevant to a requirements task. Better query formulation and integration between requirements specification and creativity support components has the potential to provide strong support for requirements tasks.

Acknowledgements

Support for combinFormation development is provided by National Science Foundation grants IIS-735897 and IIS-0747428.

References

1. Baddeley, A.D.: Human memory: Theory and practice. Lawrence Erlbaum Associates, Hove (1990)
2. Fischer, G., Giaccardi, E.: Sustaining Social Creativity. Communications of the ACM 50(12), 28–29 (2007)
3. Floyd, C., Mehl, W.-M., Reisin, F.-M., Schmidt, G., Wolf, G.: Out of Scandinavia: Alternative Approaches to Software Design and System Development. Human-Computer Interaction 4(4), 253–350 (1989)

4. Greene, S.L.: Characteristics of Applications that Support Creativity. Communications of the ACM 45(10), 100–104 (2002)
5. Henderson, D.A., Card, S.K.: Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. ACM Transactions on Graphics 5(3), 211–243 (1986)
6. Horvitz, E.: Principles of Mixed-Initiative User Interfaces. In: Proc. ACM CHI 1999, pp. 159–166 (1999)
7. Interface Ecology Lab, combinFormation (2007), http://ecologylab.net/combinFormation/
8. Kerne, A., Koh, E.: Representing Collections as Compositions to Support Distributed Creative Cognition and Situated Creative Learning. New Review of Hypermedia and Multimedia 13(2), 135–162 (2007)
9. Kerne, A., Koh, E., Smith, S.M., Webb, A., Dworaczyk, B.: combinFormation: Mixed-Initiative Composition of Image and Text Surrogates Promotes Information Discovery. ACM Transactions on Information Systems (TOIS) 27(1), 1–45 (2008)
10. Koh, E., Kerne, A., Damaraju, S., Webb, A., Sturdivant, D.: Generating views of the Buzz: Browsing popular media and authoring using mixed-initiative composition. In: Proc. ACM Multimedia, pp. 226–237 (2007)
11. Maiden, N., Robertson, S., Gizikis, A.: Provoking Creativity: Imagine What Your Requirements Could be Like. IEEE Software 21(5), 68–75 (2004)
12. Maiden, N.A.M., Ncube, C., Robertson, S.: Can Requirements Be Creative? Experiences with an Enhanced Air Space Management System. In: Proceedings 28th International Conference on Software Engineering ICSE, pp. 632–641. ACM Press, New York (2007)
13. Mamykina, L., Candy, L., Edmonds, E.: Collaborative Creativity. Communications of the ACM 45(10), 96–99 (2002)
14. Mavin, A., Maiden, N.A.M.: Determining Socio-Technical Systems Requirements: Experiences with Generating and Walking Through Scenarios. In: Proc. IEEE Requirements Engineering, pp. 213–222 (2003)
15. Mich, L., Anesi, C., Berry, D.M.: Requirements Engineering and Creativity: An Innovative Approach Based on a Model of the Pragmatics of Communication. In: Proceedings REFSQQ 2004 Workshop, Riga (2004)
16. National Science Foundation Workshop Report Creativity Support Tools, Washington, DC, June 13-14 (2005), http://www.cs.umd.edu/hcil/CST/report.html (retrieved, 02/12/2008)
17. Nuseibeh, B.A., Easterbrook, S.M.: Requirements Engineering: A Roadmap. IEEE Computer Society Press, Los Alamitos (2000)
18. Nguyen, L., Carroll, J.M., Swatman, P.A.: Supporting and Monitoring the Creativity of IS Personnel During the Requirements Engineering Process. In: Proc. Hawaii Int'l Conf. Systems Sciences (HICSS-33). IEEE Computer Society Press, Los Alamitos (2000)
19. Obsorn, A.F.: Applied Imagination: Principles and Procedures of Creative Problem Solving. Charles Scribener's Sons, New York (1953)
20. Robertson, J.: Eureka! Why Analysts Should Invent Requirements. IEEE Software, 20–22 (July/August 2002)
21. Schneiderman, B.: Creativity Support Tools – Accelerating Discovery and Innovation. Communications of the ACM, 20–29 (December 2007)
22. Sternberg, R.J. (ed.): Handbook of creativity. Cambridge University Press, Cambridge (1999)

A Quantitative Assessment of Requirements Engineering Publications – 1963-2008

Alan Davis and Ann Hickey

University of Colorado at Colorado Springs,
College of Business,
PO Box 7150,
Colorado Springs, CO 80933-7150, USA
{adavis,ahickey}@uccs.edu

Abstract. [Context and motivation] Two years ago, the authors conducted an extensive meta-analysis of the requirements engineering (RE) literature and reported a demographic analysis by date, type, outlet, author, and author affiliation for just over 4,000 RE publications. We have now added two more years and 1,200 more publications. **[Question/problem]** The current paper continues this analysis to see if the same publication trends in RE continue or if unique new trends are emerging. It explores the past ten years in more depth, and separately analyzes the trends in journals. **[Principal ideas/results]** The study uncovers some continuing trends: (1) European Union countries continue to be the leaders in publishing RE papers, (2) the UK continues to surpass most countries in annual production, (3) the USA continues to lose market share, and (4) the same institutions lead the effort. But some new trends emerge as well: (1) total production of papers in RE has decreased since its high in 2005, (2) the average number of authors per paper has increased, (3) non-RE-specific conferences and non-RE-specific conferences have published fewer RE papers, and (4) some institutions strong in RE paper production in general are not as productive with respect to journal articles, and vice versa. **[Contribution]** This paper enables RE researchers to understand where RE research is being conducted and where results are being published. Although we report some interesting trends, the data cannot help us understand causes of these trends.

Keywords: requirements engineering, requirements management, elicitation, literature analysis, specification, research analysis.

1 Introduction

Two years ago, the authors provided a demographic analysis of 44 years worth of requirements engineering (RE) publications [1]. In the current paper, we update this corpus with two more years of data and discover specific trends that have occurred in the recent past. Although two years might seem like a short time-frame to revisit these trends given the overall 46 history of RE publications, the 30% increase from 4,000 to 5,200 publications encourages this re-analysis. Moreover, since the detailed demographic basis was established in [1], this paper expands its analysis in two critical areas

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 175–189, 2009.

of interest to RE researchers. First, in addition to overall demographic trends from 1963-2008, this paper focuses on trends in the last 10 years. Second, it analyzes and compares trends for all RE publications to those for journal papers.

2 Research Method

We conducted a meta-research analysis of the full corpus of 5,200 papers published in the field of requirements engineering. Full details of our research method can be found in [1]. Briefly, we have been cataloguing RE papers for 20 years and storing key data about these publications in an Access database: paper title, publication outlet, year, all author names and their affiliations. We then constructed database queries to improve our understanding of trends in these publications.

2.1 Research Questions

In this study we deal with all publications that concern concepts of gathering, defining, pruning, and documenting requirements, needs, desires, etc. We will use the expression "requirements engineering" to refer to any or all of these concepts.

Our research questions are:

- How many publications have been written in the field of RE, and how has the production changed over the years?
- Have the most popular outlets for RE publications changed?
- How many different authors are responsible for the RE publication output? How many authors team together to author papers? Are many new authors emerging, or is the population of authors remaining fairly static?
- What countries and organizations have been responsible for writing RE papers? Has this changed significantly in the recent past? Has the distribution among government, academic, and commercial affiliations changed recently?
- Do answers to any of the above change when limited to only journals?

2.2 Data Collection

Two years ago, the database consisted of 4,089 publications gathered over the preceding 18 years. Stored in a database and publicly accessible in html via `www.uccs.cdu/faculty/adavis/reqbib.htm`, the website [2] has now been visited 60,000 times, and is used regularly by researchers all over the world. The database now includes:

- 5,198 RE publications spanning the years from 1963 through 2008.
- We located full source or abstracts for approximately 4,565, or 88%.
- We determined the complete list of authors for 5,194, or 99.9%.
  - 5,973 unique authors of RE publications
  - 11,988 unique assignments of authors to publications, and we determined affiliations of 11,675, or 97.4%, of the author-pub pairs.

Although we make no completeness claims, we believe the database is (a) the largest ever collected, and (b) complete enough to make the data analysis trends accurate.

3 Results for All Requirements Publications

3.1 Publication Volume and Trends

The RE research area had exhibited exponential growth between 1977 and 2006 [1], but that growth has leveled out more recently. Fig 1 shows the growth during the past 46 years in 5-year increments. For consistency with the format of the data we used in the 2007 study, we have linearly extrapolated RE publication production for 2007-2011 based on the papers recorded for 2007 and 2008. To better understand more recent trends, Fig 2 shows the past 10 years. We see a noticeable decrease in production in the past 3 years. We do not fully understand its reasons, but have reached some preliminary conjectures, to be discussed later in this paper.

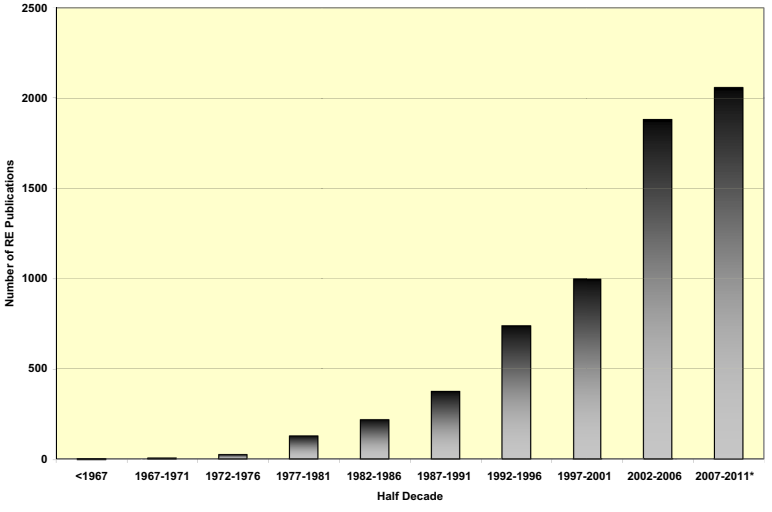


Fig. 1. Quantity of Publications in Domain of Requirements Engineering

3.2 Publication Types and Outlets

We investigated outlets of all publications in our database. Fig 3 shows how the number of publications for each outlet type (the figure shows two types of conference papers – regular and auxiliary – where auxiliary papers include introductions, panel reports, etc.; and two types of journal papers –regular and auxiliary – where auxiliary papers include guest editor introductions, columns, abstracts, etc.) has changed over the past 10 years. Notice that the percent of RE papers published in conferences has increased slowly from around 50% to over 60%. Journal papers have remained fairly steady at 20-25%, with a decrease in its relative contribution since 2007. The number of books published has decreased since the heyday in 1999-2005. Further analysis of specific outlets highlights the leading venues for RE publications; Fig 4 shows all venues accounting for 2% or more, and highlights how their popularity has changed during the past two years. Notice that the IEEE Requirements Engineering Conference and REFSQ have grown in

proportion to the general growth of RE publications, the relative size of the *Requirements Engineering Journal* has decreased slightly, and the INCOSE Symposium on Systems Engineering has decreased its emphasis on RE.

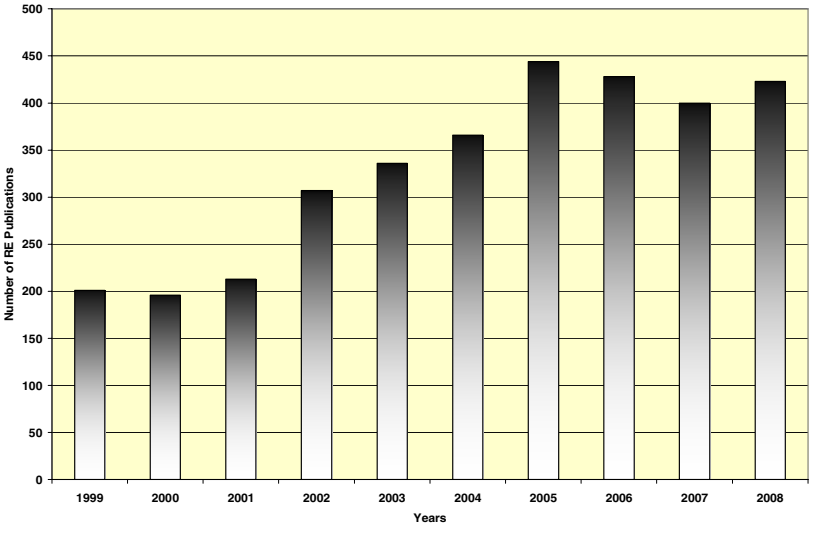


Fig. 2. Quantity of Publications in Domain of RE (past 10 years)

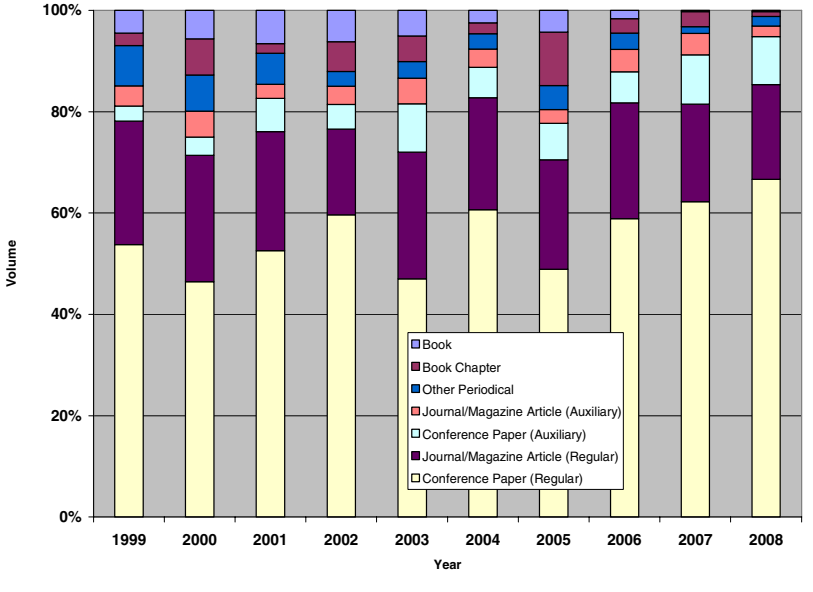


Fig. 3. Publication Outlets by Year

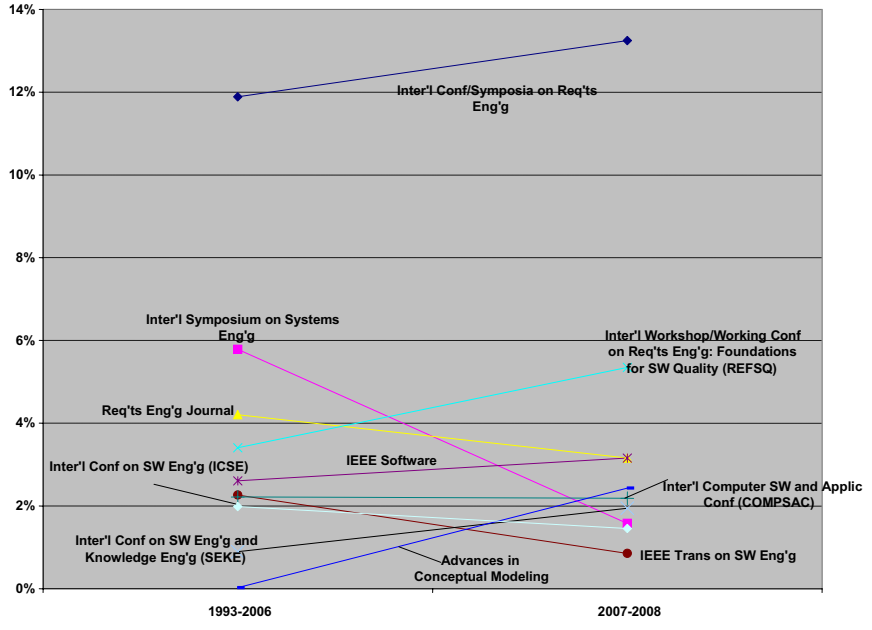


Fig. 4. Leading RE Publication Venues

3.3 Author Productivity and Authorship Patterns

We also investigated the authors of RE publications. The 5,198 publications were written by a total of 5,973 different individuals. We analyzed the average number of authors on each paper, as shown in Fig 5; here we see a distinct increase from an average of 2 authors per paper to an average of 3 authors per paper, over a period of just 10 years. This is, in fact, a long-term trend; the average number of authors per paper published prior to 1999 was below 2. We also wanted to discover if requirements engineering research is becoming more insular, so we analyzed how many authors each year are new[1] authors. The results, shown in Fig 6, show a phenomenal trend, i.e., that very few new authors are emerging. Ten years ago, 40% of all authors of RE papers were new to the field; today, only 10% are new to RE. Are we becoming more insular? Does our conference and journal review process contribute to this? Would this change if we used a double-blind review process 2 (common in most disciplines)? Are the members of all are review committees composed of the same group of individuals? Is our field simply not attracting new researchers because of a decrease in interest? Or is this phenomenon commonplace for any new field?

3.4 Author Affiliations

We then placed each publication into one or more of four categories based on affiliations of its authors: academic, industry, government, and unknown. To no surprise a

[1] Defined as "did not author an RE paper in the preceding 5 years."

large majority of RE papers are written by academics. But as can be seen in Fig 7, a growing majority (increasing from 73% to 87% during the past ten years) of RE papers have at least one author affiliated with an academic institution, and a shrinking minority (decreasing from 31% to 27% during the past ten years) of RE papers have at least one author affiliated with a commercial institution.

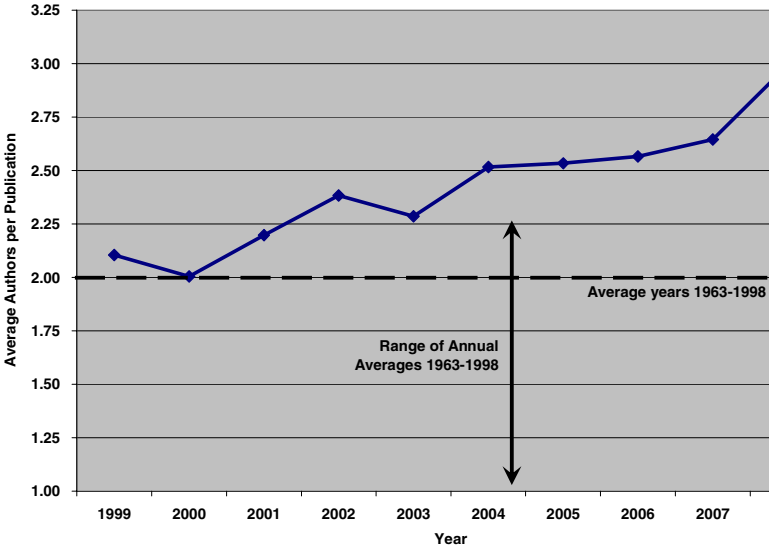


Fig. 5. Average Number of Authors per Publication

Author affiliations represent over 1,541 different organizations worldwide. The top 20 RE publication-producing organizations, along with their relative ranking, and total number of publications are shown in Table 1[2]. It is interesting to note that 3 of the top 10 are in UK (including 3 of the top 4), 2 are in Canada, and 1 is in each of France, Germany, Netherlands, Sweden and USA. Just below the 1% cutoff are Pontifícia Universidade Católica do Rio de Janeiro, and Technischen Universität München. To better understand how the top five organizations have produced RE publications recently, Fig 8 shows their annual production (three year moving averages of a percent of papers produced that year) for the past 10 years. Notice that #1, City University of London, had a lull in the early 2000's, but is now producing over 3% of every year's RE paper production. University of Toronto and Manchester University have remained consistent for the past ten years, producing roughly 2.5% and 1% of papers annually, respectively. The #3 organization, Lancaster University also experienced a lull in the early 2000's but now produces 2% annually. Fraunhofer had produced more in the early 2000's but has now slowed down.

[2] Note that small publication counts separate all but the top 3; thus inadvertent omissions by the researchers can have a significant affect on rankings.

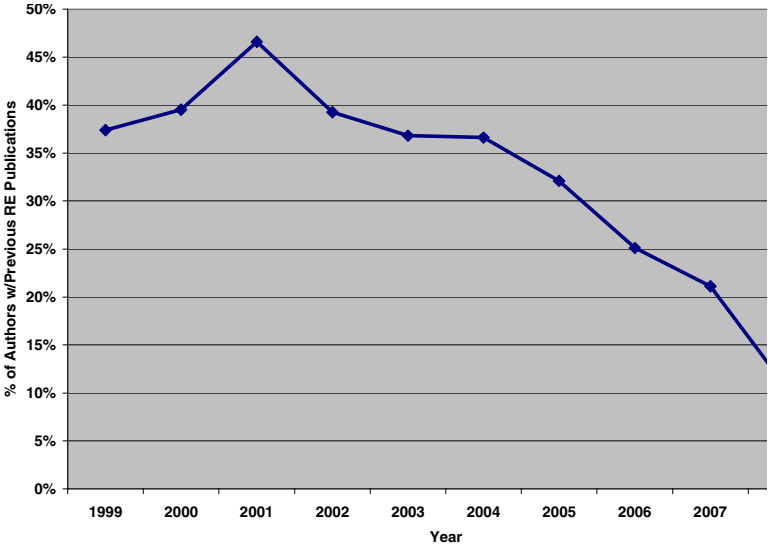


Fig. 6. New RE Authors

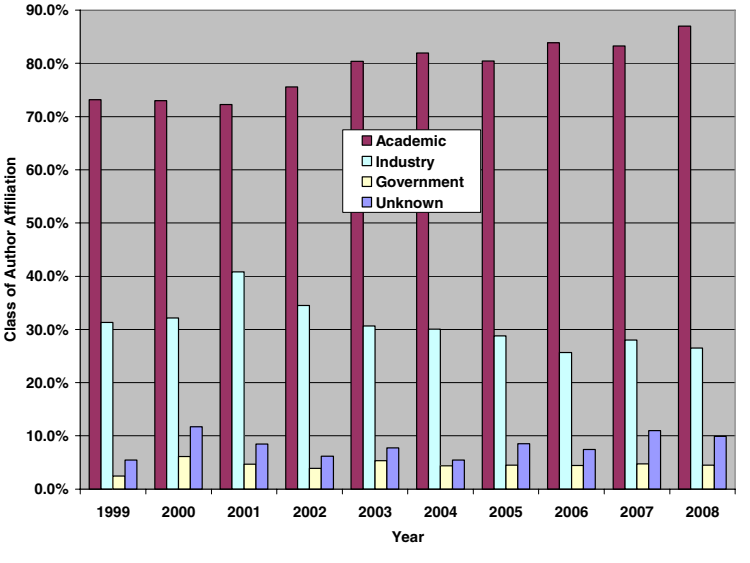


Fig. 7. Percent of RE Authors from Classes of Affiliation[3]

To see how individual countries have contributed, let us look at a ranked list of all countries that have produced 1% or more of all RE publications as of 2008, as shown in

[3] Note that the sum of the percents for any one year exceed 100% because some papers include multiple authors from multiple classes of organizations.

Table 2. Meanwhile, Table 3 combines the European Union (EU) countries. Finally, Fig 9 shows the ten year trend for the top 7 countries; this figure makes it clear that the EU has increased its share in the past ten years, while US has lost its share.

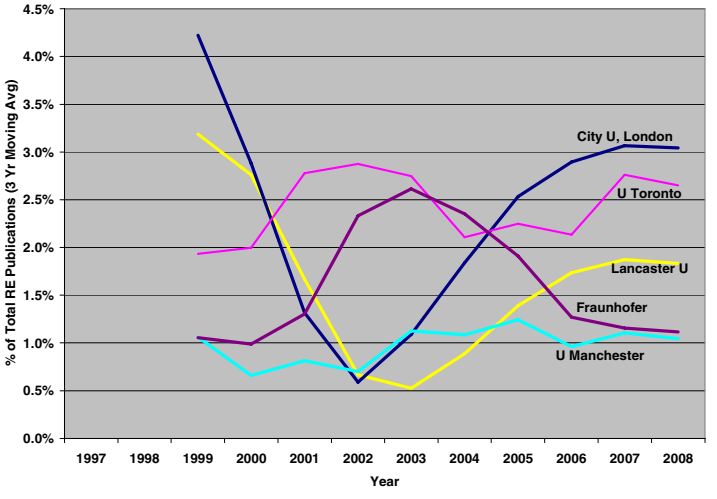


Fig. 8. Top Five RE-Publishing Organizations – Last 10 Years (3 Yr Moving Averages)

Table 1. Top 20 Organizations

Organization	Rank	# Pubs	% of Total
City U, London	1	127	2.44%
U Toronto	2	112	2.15%
Lancaster U	3	76	1.46%
U Manchester	4	61	1.17%
Fraunhofer - Germany	4	61	1.17%
U Calgary	6	59	1.14%
U Colorado, Colorado Springs	7	56	1.08%
U Twente	8	54	1.04%
Lund U	9	52	1.00%
U Paris 1, Panthéon-Sorbonne	9	52	1.00%
U Technology, Sydney	11	51	0.98%
Open U	12	50	0.96%
Imperial College	13	48	0.92%
Jet Propulsion Lab	13	48	0.92%
AT&T - USA	15	46	0.88%
U Southern California	16	44	0.85%
U Waterloo	17	43	0.83%
George Mason U	18	41	0.79%
RWTH - Aachen U of Technology	19	40	0.77%
IBM - USA	20	39	0.75%

Table 2. Top 19 Countries

Country	Rank	# Pubs	% of Total
USA	1	1966	37.8%
UK	2	814	15.7%
Canada	3	395	7.6%
Germany	4	343	6.6%
Australia	5	250	4.8%
Spain	6	190	3.7%
Italy	7	163	3.1%
Sweden	8	156	3.0%
Japan	9	147	2.8%
France	10	146	2.8%
Netherlands	11	138	2.7%
China	12	120	2.3%
Brazil	13	110	2.1%
Austria	14	93	1.8%
Belgium	15	89	1.7%
Finland	16	61	1.2%
Israel	17	57	1.1%
India	18	56	1.1%
Korea, South	18	56	1.1%

Table 3. Top 10 Countries (EU Considered as One)

Country	Rank	# Pubs	% of Total
EU	1	2397	46.1%
USA	2	1966	37.8%
Canada	3	395	7.6%
Australia	4	250	4.8%
Japan	5	147	2.8%
China	6	120	2.3%
Brazil	7	110	2.1%
Israel	8	57	1.1%
India	9	56	1.1%
Korea, South	9	56	1.1%

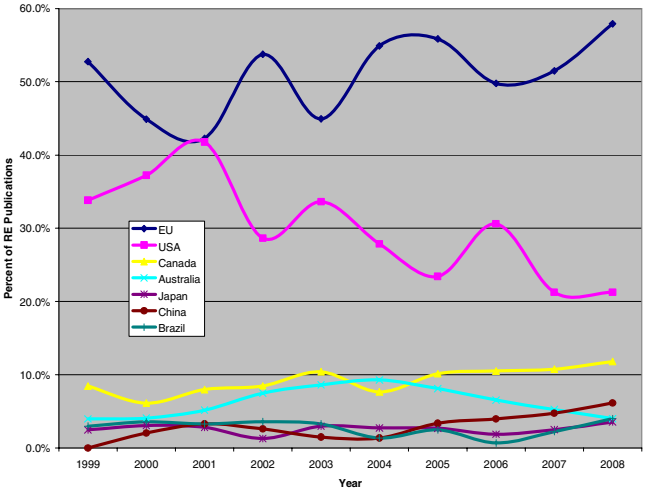


Fig. 9. Top Seven RE-Publishing Countries/Unions – Last 10 Years

4 Results for Journal Requirements Publications

Recognizing that many academic institutions emphasize publishing in journals rather than other outlets, this section analyzes the subset of the entire RE publication corpus

corresponding to papers in journals; it does not include auxiliary papers such as guest editor introductions, columns, editorials, book reviews, and so on.

4.1 Publication Volume and Trends

Until 2007, growth in RE journal publication closely paralleled the growth in RE publications in general (contrast Fig 10 with Fig 1). As before, we have linearly extrapolated RE publication production for 2007-2011 based on the papers for 2007 and 2008. However, Fig 11, which shows detailed data for just the past 10 years, exhibits a major drop in journal publications in the past 2 years. The reasons for this are unclear, but may in part be due to: (a) requirements engineering may have fallen out of vogue as a serious research topic, (b) fewer journal outlets accepted RE publications (e.g., in 2005 and 2006, approximately 50 journals published RE papers, but in 2007 and 2008, approximately 40 did so; the major RE journal publishers did not decrease their production, but fewer related journals published papers on RE). This trend needs to be watched in coming years to see if it continues.

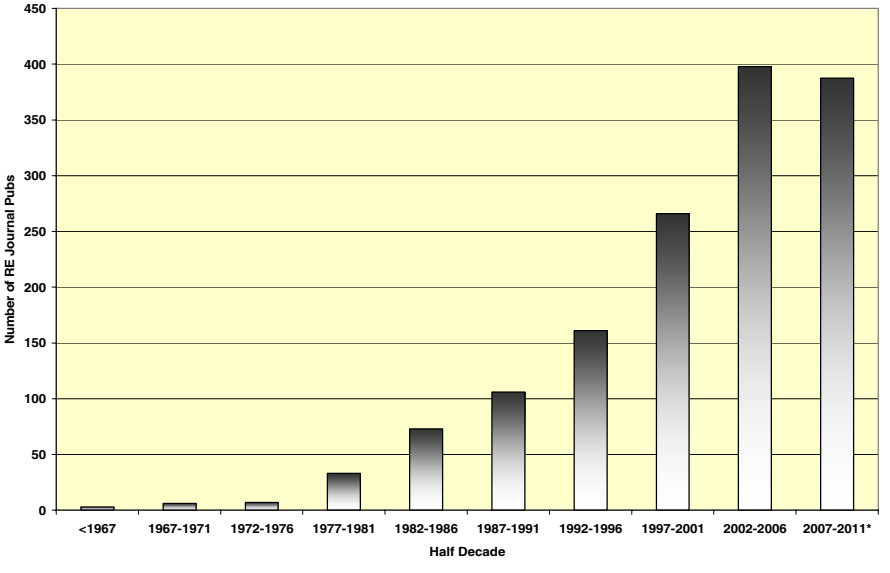


Fig. 10. Quantity of Journal Publications in Domain of RE

4.2 Author Productivity and Authorship Patterns

The 1,209 journal papers were written by a total of 1,980 different individuals. To better understand this, we again analyzed the average number of authors on each paper, as shown in Fig 12 (this shows the same increasing trend as for RE publications in general, as shown in Fig 5). We also wanted to discover if requirements engineering research is becoming as insular with journal publication as it is with RE

publications in general, so we analyzed how many authors of journal papers each year are new authors (i.e., did not publish *any* RE paper in the preceding 5 years). The results, shown in Fig 13, show the trend in journals is even more insular than in general, with only 8% of papers being written by first-time authors. Although one could argue that journal papers are usually written by individuals who have published before, the single-digit trend does not go back more than two years; prior to 2006, between 15% and 40% of the RE journal articles were written by first-timers.

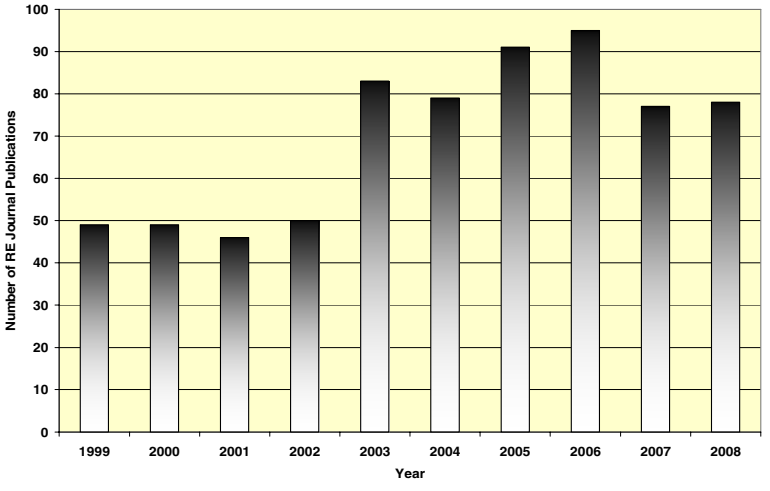


Fig. 11. Quantity of Journal Publications in Domain of RE (past 10 years)

4.3 Author Affiliations

We categorized each journal publication by affiliations of its authors. Fig 14 shows the results. To no surprise given academic pressure to publish in more prestigious outlets, a greater percentage of journal article authors (than RE authors in general) are affiliated with academic institutions. Affiliations for authors of RE journal papers include 802 different organizations worldwide. Table 4[4] contains a list of all organizations responsible for 1% or more of the total RE journal publication, along with their ranking as of 2008, and total number of journal publications. Just below the cutoff for this table are University College of London, Lunds Universitet, and Pontifícia Universidade Católica do Rio de Janeiro.

Some interesting results emerge when comparing Table 4 to Table 1: Some organizations that were not highly ranked for RE publications in general are much more impressive when we examine just journals: University of Minnesota, Minneapolis; University of Maryland. College Park; Georgia State University; Naval Research Lab; Brunel University; Naval Postgraduate School; and Universidad Politécnica de

[4] Note that small publication counts separate all but the top 2; thus inadvertent omissions by the researchers can have a significant affect on rankings.

Madrid. Other institutions, which fared well with respect the RE publications in general, did not make the list of top journal producers: Fraunhofer, Universiteit Twente, Lunds Universitet, Université de Paris 1, University of Technology, Sydney, Open University, University of Waterloo, George Mason University and IBM-USA.

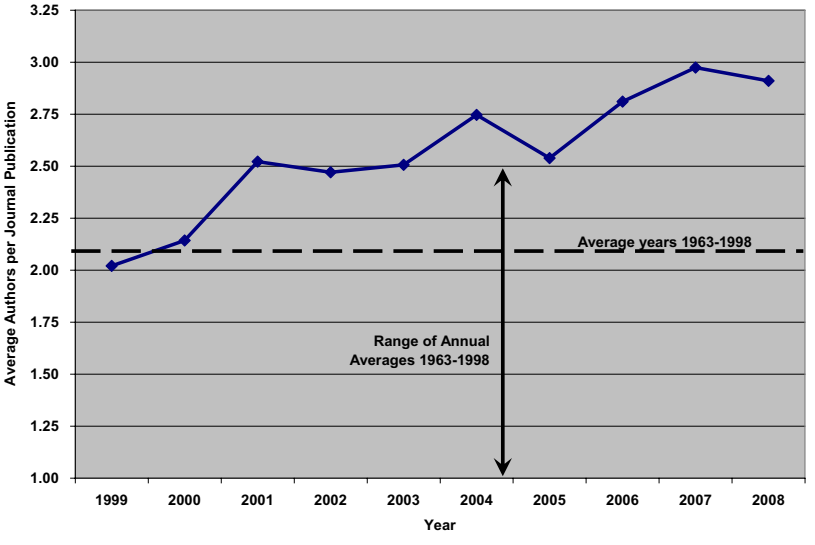


Fig. 12. Average Number of Authors per Journal Publication

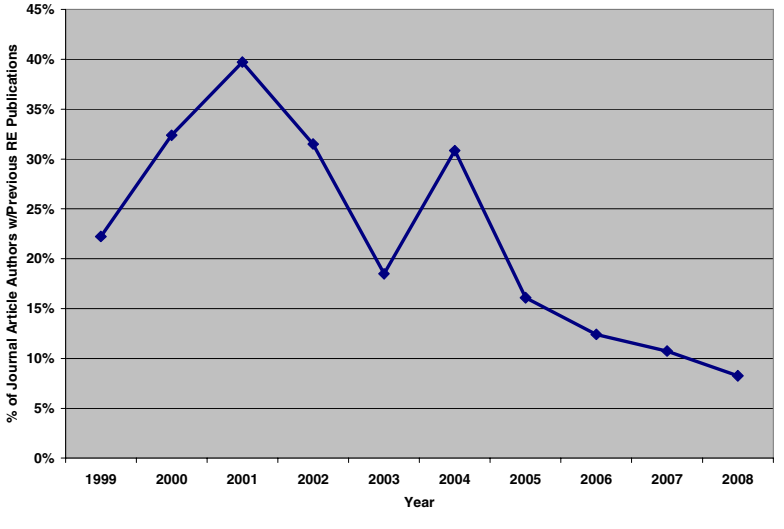


Fig. 13. New RE Journal Authors

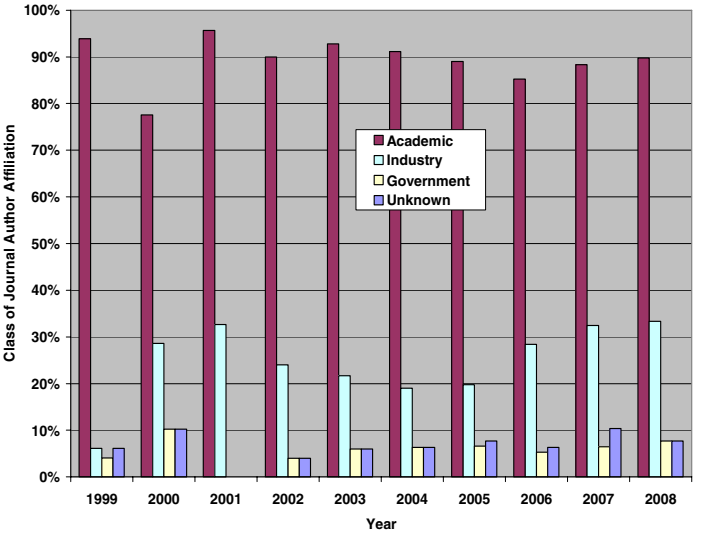


Fig. 14. Percent of RE Journal Authors from Classes of Affiliation[5]

Table 4. Top RE Journal Producing Organizations

Organization	Rank	# Pubs	% of Total
City U, London	1	33	2.76%
U Manchester	2	25	2.09%
U Toronto	3	19	1.59%
AT&T - USA	4	18	1.51%
U Colorado, Colorado Springs	5	17	1.42%
U Calgary	6	16	1.34%
Lancaster U	6	16	1.34%
U Minnesota, Minneapolis	6	16	1.34%
U Maryland, College Park	9	15	1.26%
Georgia State U	10	14	1.17%
U Southern California	10	14	1.17%
Naval Research Lab	12	13	1.09%
Jet Propulsion Lab	12	13	1.09%
Brunel U	12	13	1.09%
Naval Postgraduate School	15	12	1.01%
RWTH - Aachen U of Tech	15	12	1.01%
U Politécnica Madrid	15	12	1.01%
Imperial College	15	12	1.01%

To see how individual countries have contributed to this trend, let us look at a ranked list of all countries that have produced 1% or more of all RE journal publications as of the end of 2008, as shown in Table 5. Meanwhile, Table 6 combines the EU. Notice the

[5] Note that the sum of the percents for any one year exceed 100% because some papers include multiple authors from multiple classes of organizations.

top 7 countries for journal production are identical to the top 7 countries for RE production in general. In fact, the only significant difference between Table 3 and Table 6 is that Taiwan emerges on the list of leading journal-producers. Finally, Fig 15 shows the 10 year trend for the top 7 countries. Once again, the increasing role of EU and decreasing role of USA is apparent. This is interesting contrast to the observations by Lyytinen, et al [3], who report that Europe is behind USA in publishing of top-quality journal papers in information systems in general. This difference in results could be the caused by the fact that Lyytinen, et al., (a) studied IS in general, and we are studying RE specifically, (b) may not consider RE as part of IS, or (c) examined only the top IS journals, whereas we studied all journals.

Table 5. Top RE Journal-Producing Countries

Country	Rank	# Pubs	% of Total
USA	1	479	39.7%
UK	2	227	18.8%
Canada	3	93	7.7%
Germany	4	64	5.3%
Australia	5	51	4.2%
Spain	6	44	3.6%
Sweden	7	40	3.3%
Italy	8	39	3.2%
Netherlands	9	34	2.8%
Japan	10	31	2.6%
France	11	25	2.1%
Taiwan	12	24	2.0%
China	13	23	1.9%
Israel	13	23	1.9%
Korea, S.	15	22	1.8%
Belgium	16	20	1.7%
Austria	17	19	1.6%
Brazil	17	19	1.6%
Denmark	19	17	1.4%
India	20	14	1.2%

Table 6. Top RE Journal-Producing Countries (EU Considered as One)

Country	Rank	# Pubs	% of Total
EU	1	574	47.5%
USA	2	479	39.7%
Canada	3	93	7.7%
Australia	4	51	4.2%
Japan	5	31	2.6%
Taiwan	6	24	2.0%
China	7	23	1.9%
Israel	7	23	1.9%
Korea, S.	9	22	1.8%
Brazil	10	19	1.6%
India	11	14	1.2%

5 Summary and Conclusions

Overall interest in research in requirements engineering may be leveling off and certainly appears to be slowing from the exponential growth reported in our previous analysis [1]. Many trends observed previously continue including the European Union and UK leadership in RE publications, USA's continuing loss of market share, and the institutions who lead in RE publications. The new analysis focusing on RE journal publications highlights many of these same trends, with the only major differences appearing in the list of leading institutions. New emerging trends are even more interesting. The decrease of RE publications in 2007 and 2008 should be carefully monitored. The dramatic increase in the number of authors per paper is also surprising. However, the most surprising results arose from the focus on the last 10 years. The new dependence on RE-specific publication outlets, the increasing dominance of academic vs. industrial authors, and, most critically, the decreasing numbers of new authors may indicate some combination of an increasing insularity of

the RE research community and/or a decreasing interest in RE research. The inability to definitively identify the cause of these trends is the major limitation of this research, but either possibility is worrisome. Future research will focus on close monitoring and more in-depth analysis of these trends to see if they persist.

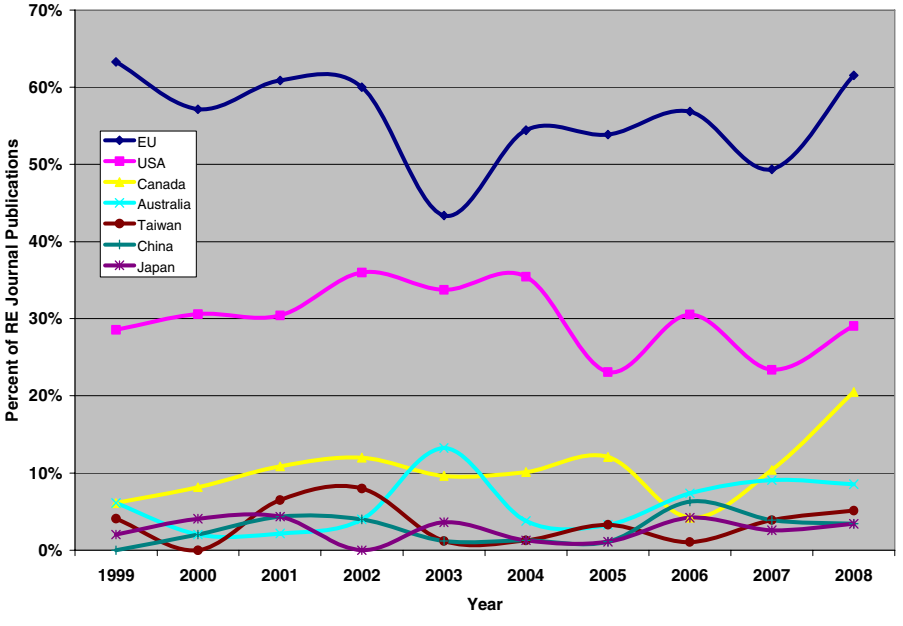


Fig. 15. Top Seven RE Journal- Producing Countries – Last 10 Years

References

1. Davis, A., Hickey, A., Dieste, O., Juristo, N., Moreno, A.: A quantitative assessment of requirements engineering publications – 1963-2006. In: Sawyer, P., Paech, B., Heymans, P. (eds.) REFSQ 2007. LNCS, vol. 4542, pp. 129–143. Springer, Heidelberg (2007)
2. `http://www.uccs.edu/faculty/adavis/reqbib.htm`
3. Hill, S., Provost, F.: The myth of the double-blind review?: Author id using only citations. ACM SIGKDD Exp. News 5(2), 179–184 (2003)
4. Lyytinen, K., Baskerville, R., Iivari, J., Te'eni, D.: Why the old world cannot publish? Overcoming challenges in publishing high-impact IS research. Euro. J. Info. Sys. 16(4), 317–326 (2007)

Assurance Case Driven Case Study Design for Requirements Engineering Research

Robin A. Gandhi[1] and Seok-Won Lee[2]

[1] University of Nebraska at Omaha, College of Information Science and Technology, 6001 Dodge Street, Omaha, NE 68182
rgandhi@unomaha.edu
[2] University of North Carolina at Charlotte, College of Computing and Informatics, 9201 University City Blvd., Charlotte, NC 28223
seoklee@uncc.edu

Abstract. [Context and motivation] Case studies have the potential to be an essential bridge between the constructive (build new theories, algorithms, or methods to address practical problems) and the empirical (develop evidence through observation of or experience with existing methods or artifacts in practice) approaches to requirements engineering research. **[Question/problem]** To realize this potential, our aim is to provide representational guidance for designing a case study as a part of the invention process of a novel requirements engineering methodology (REM). **[Principal ideas/results]** In this paper, we present the innovative use of assurance cases to emphasize argumentation and rigorous evidence planning during case study research design. **[Contribution]** The steps involved in case study research design using the assurance case notation are outlined as a systematic way to plan a validation effort for a REM.

Keywords: Case study, Assurance Case, Requirements, Research methods.

1 Introduction

A Requirements Engineering Methodology (REM) provides analytical capabilities to Subject Matter Experts (SMEs) who execute its steps for eliciting, modeling, analyzing, negotiating or validating requirements. Well-defined usage, representation and measurement of these capabilities and related artifacts are essential to demonstrate the achievement of the desired outcomes from the REM as well as their continuous improvement. An outcome is defined as a statement that describes what a SME is expected to know and to be able to perform by using the REM or the expected quality of resulting artifacts. From the definition, it is clear that any REM validation effort is heavily dependent on the skills of SMEs who participate in the validation exercise, and the problem domain.

For a newly invented REM, the outcomes are difficult to evaluate as no materializations may exist yet in the real-world context and the SMEs may have no prior experience or training for the techniques required to be applied. Comparison with other existing REMs and benchmarking is not meaningful as the unique nature

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 190–196, 2009.

(philosophy, models, steps, and techniques) of each REM influences human analytical reasoning and resulting artifacts differently. Under these circumstances, a defined strategy for data collection and analysis is necessary to make valid inferences about the fitness/quality of the desired REM outcomes from the observed phenomena or artifacts in the real world context. To carry out this effort, case study research designs allow an investigator to clearly identify the followings [12] [8] [10]: 1) The research questions; 2) The REM outcomes being studied; 3) What are the resources to be examined and what data to collect; 4) What are the logics linking the data to the REM outcomes; and 5) What are the criteria for interpreting the findings.

If used correctly, case study as a validation strategy reveals: why and how a newly invented REM works; the effectiveness of the stated outcomes; and opportunities for further improvement. However, planning a case study is a significant undertaking. Yin [12] points out that a good case study is difficult to perform, and we hardly understand the required skills on the part of the investigators or reviewers dealing with case studies. Vague or implicit research designs will only add to the many misunderstandings and skepticism [5] already widespread about case studies. Additionally, if the case study design is an afterthought the newly invented REM may not provide optimal opportunities for data collection necessary to address the original research questions. The REM steps may also lack traceability and rationale to support the expected outcomes if such contribution is not clearly identified in advance.

It is clear that systematic guidance while building case study research designs is essential. Particularly, we emphasize the need for representational mechanisms that guide and assure the development of a rigorous case study research design during the REM invention itself. This approach can effectively combine the constructive component of REM development with an empirical component to validate REM outcomes in the real world context. To facilitate this goal, in the following sections we describe the steps in combining *assurance cases* [7] and case study research designs in requirements engineering research. These steps have been developed in the context of planning a case study while inventing a novel REM. While the complexity of planning data collection and analysis in context with SMEs makes our approach particularly suited for requirements engineering research, the steps are generally applicable and can be extended for planning case studies in different domains.

2 Representing Assurance Case and Case Study Design

Often compared to a structured legal argument, an assurance case [7] is a self-contained information unit that includes a top-level goal/claim, an argument in the form of continuous claim refinement until the sub-claims can be specified in operational terms, and evidence that the claims have been satisfied. Its purpose is to provide a clear, comprehensive and defensible argument that the claimed objectives are achieved in a certain context. On the other hand, the case study evaluation strategy is mostly appropriate for “how” and “why” types of study questions [12]. These question types also correspond to top-down and bottom-up traversals in a goal refinement hierarchy, respectively. As a result, the goal structuring notation [7] of assurance cases provides a natural fit for representing case study research designs.

With an early start to case study design using assurance cases, our rationale is to identify more intuitive sources of evidence by exploring the possibilities of instrumentation (i.e. opportunities for evidence generation and gathering) as each step of the REM is being developed. Early planning of evidence generation and collection alongside the explication of the theoretical propositions that underlie the REM will assure that the identified outcomes are in fact being addressed by the methodological steps and measured properly in the case study research design. To facilitate this process, the five case study design components can be systematically captured in the form of a graphical assurance case notation as Fig. 1. The assurance case notation [7], as shown in Fig. 1, consists of the following elements: goal/claim nodes, strategy nodes, context nodes, solution/evidence nodes, and most importantly the interconnections among these nodes to construct a valid and convincing argument. Strategy elements clarify the nature of an argument, while context elements specify the conditions under which a goal/claim is stated.

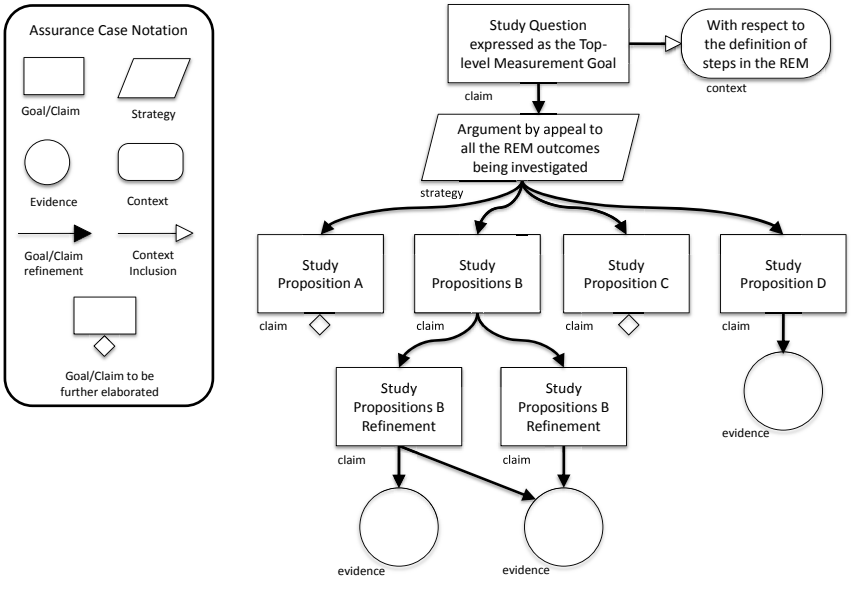


Fig. 1. An Assurance Case Skeleton and its relevance to Case Study Design Components

For developing case study research designs using assurance cases alongside the design of a new REM, a step by step method is outlined as follows:

Step 1: Study Question and the Top-level Measurement Goal/Claim. To identify opportunities for REM instrumentation as early as possible, the primary study question ("why" and "how" a newly invented REM works) needs to be re-written as the top-level measurement claim/goal in the assurance case notation. This conversion in some sense starts the linking process between constructive and empirical research approaches. For the assurance case to follow a logical argument, a claim is always worded as a predicate (i.e. it can only be true or false).

Step 2: Propositions and Goal/Claim Refinement. Case studies benefit from the prior development of theoretical propositions to guide data collection and data analysis [12]. These propositions are assertions that determine what should be examined within the scope of the study. Typically these assertions emerge from all the desired REM outcomes that reflect the theoretical propositions that guide the REM development. To capture this general rationale in an assurance case, the Strategy modeling element suggests the form that an argument is going to take for satisfying the top-level claim. Specific theoretical propositions then form the sub-claims that refine the top-level claim as shown in Fig. 1. The REM progression may correspond to the levels of intellectual behavior [3] of the SMEs or related artifacts.

Step 3: Units of Analysis and Case boundary. With the REM itself being the "case", the units of analysis include the steps in the methodology, the models and methods used in each step, and the characteristics of the methodology itself. To this end, the assurance case claims and steps in the REM clearly establish the case boundary and identify the relevant phenomena/artifacts.

Step 4: Linking Logic and Goal/Claim Operationalization. The theoretical propositions may exist at a level of abstraction such that they cannot be measured directly. Through a goal refinement process, assurance cases can effectively explicate the rigor adopted by the investigator in linking the high level theoretical propositions to concrete measurement claims/goals that can be specified in operational terms of the REM steps. A particularly effective and disciplined way to gather and organize case study evidence is to construct an ordered set of questions with respect to the steps in the REM as a Summary Sheet [8]. For a newly invented REM, the summary sheet combined with the given instructions (e.g. a tutorial or workshop to teach or educate the REM) prior to a case study helps SMEs to identify the units of analysis and the corresponding evidence (qualitative or quantitative) that needs to be captured towards the case study propositions. The summary sheet columns in a table format, as shown in Fig. 2, allows SMEs to effectively conduct the steps in the REM and capture evidence with no interventions required from the investigator. In the fourth column, the identification of assurance leaf-node claims for each row of the summary sheet clearly explicates what evidence from the unit of analysis supports/rejects the claims.

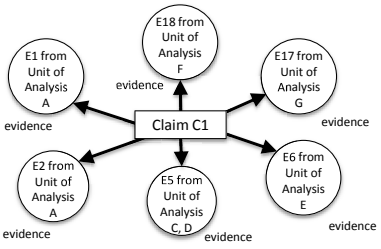
Unit of Analysis	Evidence to be captured by SMEs based on specific propositions	Step/Model/Method in the REM	Related Claims in the Assurance Case

Fig. 2. Elements of a Summary Sheet

Step 5: Data Analysis through Stratification. Unorganized collections of data do not produce meaningful insights. For initial guidance in selecting data analysis techniques, we have developed a general tabular structure wherein the gathered evidence is placed in the cells at the intersection of the relevant leaf-node claims (columns) of the assurance case and units of analysis (rows). The resulting presentation, as shown in Fig. 3 (a), provides a multi-dimensional categorization of the gathered evidence in the summary sheet. This representation can also be combined with more specific data analysis techniques such as structural analysis [11] or comparative gap analysis [4].

Leaf-Node Claims / Units of Analysis	Claim C1	Claim C2	. .
Unit of Analysis A	Evidence E1 Evidence E2	Evidence E2	. .
Unit of Analysis B		Evidence E3 Evidence E4	. .
.	.	.	.

(a)



(b)

Fig. 3. (a) Stratification of Evidence gathered in the Summary Sheet; (b) Convergence of Multiple Sources of Evidence to support/reject a Claim

Case study inquiry relies on multiple sources of evidence, with data needing to converge in a triangulating fashion [12]. The stratification of evidence, as shown in Fig. 3 (b), can capture multiple sources of evidence along with their relationships (converging or non-converging sources of evidence) towards making stronger analytical conclusions about a claim. An assurance case allows relatively weak evidence to be combined in argumentation with other evidence to make a stronger conclusion [6]. The evidence stratification effectively utilizes this flexibility to include varying degrees of rigor in evidence as well as different types of evidence in a case study to make appropriate analytical generalizations. This presentation exercise with summary sheet questions may also identify instrumentation defects.

Step 6: Criteria for Interpreting the Findings. As with any empirical evaluation, usually there is no precise way of setting the criteria for interpreting the case study findings. The assurance case representation adds much needed rigor to this design step. In particular, the criteria are clearly based on metrics that evaluate the importance/significance of the evidence to support the argument. These metrics can be supported by quantitative or qualitative measures depending on the types of the evidence available from the case study.

The assurance case as a whole helps to maximize the utilization of limited resources and available evidence in a given research setting. Their analytical structure allows a careful and fair review of the level of trust that can be put into the case study results. In other words, an explicit body of knowledge exists for a reader to determine the level of trust to be expressed in results that may have different values depending on the research settings [9].

Step 7: Testing for Threats to Validity and Assurance Case Representation. *Construct Validity.* The assurance case argument maintains an explicit chain of evidence from the original study question to any conclusions that are made. Such traceability enables a reviewer or participating SMEs to assess that correct operational measures have been chosen for the characteristics of the REM being studied.

External validity. This test for a case study is often addressed through replication logic in multiple case study design, which is outside our current scope.

Internal Validity. For explanatory type case study, an assurance case facilitates systematic review of the inferences being made about causality between the REM

execution and the desired outcomes. During review, an assurance case improves the chance to detect if any factor contributing to the observed outcomes is missing.

Reliability. The assurance case outlines a repeatable protocol for case study execution and data analysis. When multiple investigators are involved, the assurance case can help to promote a common understanding regarding the research design.

3 Discussion and Future Work

The work presented here builds upon the goal-oriented case study research design for a newly invented software engineering methodology by Lee et al. [8]. Identifying the representational strengths of assurance cases for the early planning of case studies during REM invention, and evidence stratification to support logical argumentation are the novel ideas contributed through this work. In the footsteps of other goal-driven approaches such as GQM [2], our selection of the assurance case notation puts equal emphasis on structured logical argumentation and planned evidence collection, which are both important elements of a case study to draw analytical generalizations. In contrast to the goal/claim, argument, and evidence structure of assurance cases, theory diagrams [1] use action-oriented structure of precondition, actions, results, and effects to model theories. In both cases, the objective is to increase reviewability through a defined logic, linking the theoretical propositions to the observed phenomena.

Our exploration of the case study design possibilities here is by no means complete. Many subtleties will emerge as the approach outlined in this paper is employed across cases and their results are reported. Our approach is intended to provide a structured capability for tracing the usages, acceptance, and feasibility of a newly invented REM based on defined data collection and analysis strategy over a period of time across multiple cases and in different domains. We hope that early adoption of this strategy, while a new REM is being developed, will reduce the gap between constructive and empirical research approaches.

An important aspect of our future work includes the codification of case study design approaches. Unlike other research strategies, a comprehensive catalog of research designs for case studies does not exist [12]. Assurance case patterns [6] provide a promising direction for codifying different case study designs to be used in requirements and software engineering methodology validation. Well-codified research designs will guide the selection of cases and units of analysis that are similar to previous works in the literature and provide points of comparison. This form of case study design is highly encouraged in social sciences.

References

1. Axelsson, K., Goldkuhl, G.: Theory Modelling - Action Focus when Building a Multi-Grounded Theory. In: 3rd Euro. Conf. on Research Methods in Buss. & Mgmt. (2004)
2. Basili, V.R., Caldiera, G., Rombach, H.D.: The Goal Question Metric Approach (1994)
3. Bloom, B.S., Krathwohl, D.R.: Taxonomy of educational objectives: The classification of educational goals. Handbook 1: Cognitive domain. Longmans, New York (1956)

4. Breaux, T., Anton, A., Dorfman, B.: Legal Requirements, Compliance and Practice: An Industry Case Study in Accessibility. In: 16th IEEE Req. Engg. Conf., Spain (2006)
5. Flyvbjerg, B.: Five Misunderstandings about Case Study Research. In: Qualitative Research Practice, pp. 420–434. Sage Publications, Thousand Oaks (2004)
6. Goodenough, J., Lipson, H., Weinstock, C.: Arguing Security - Creating Security Assurance Cases. Technical Report, Software Engineering Institute (2007)
7. Kelly, T.P., Weaver, R.A.: The Goal Structuring Notation: A Safety Argument Notation. In: DSN Workshop on Assurance Cases. Florence, Italy (2004)
8. Lee, S.W., Rine, D.C.: Case Study Methodology Designed Research in Software Engineering Methodology Validation. In: 16th SEKE Conf., Canada, pp. 117–122 (2004)
9. Shaw, M.: The coming-of-age of software architecture research. In: 23rd International Conference on Software Engineering, Toronto, Canada, pp. 657–664 (2001)
10. Sim, S.E., Perry, D.E., Easterbrook, S., Aranda, J.: Case Studies for Software Engineers. In: The 28th International Conf. on Software Engineering, Tutorial, China (2006)
11. Wasson, K.S.: A Case Study in Systematic Improvement of Language for Requirements. In: 14th IEEE International Requirements Engineering Conference, pp. 6–15 (2006)
12. Yin, R.K.: Case Study Research: Design and Methods. In: Applied Social Research Methods Series, 2nd edn., vol. 5. Sage Publications, Thousand Oaks (1994)

Translation of Textual Specifications to Automata by Means of Discourse Context Modeling

Leonid Kof

Fakultät für Informatik, Technische Universität München,
Boltzmannstr. 3, D-85748, Garching bei München, Germany
kof@informatik.tu-muenchen.de

Abstract. [**Context and motivation**] Natural language is the main presentation means in industrial requirements documents. In such documents, system behavior is specified either in the form of scenarios or in the form of automata described in natural language. The behavior descriptions are often incomplete: For the authors of requirements documents some facts are so obvious that they forget to mention them; this surely causes problems for the requirements analyst.

[**Question/problem**] Formalization of textual behavior description can reveal deficiencies in requirements documents. Formalization can take two major forms: it can be based either on interaction sequences or on automata, cf. survey [1]. Translation of textual scenarios to interaction sequences (Message Sequence Charts, or MSCs) was presented in our previous work [2,3,4]. To close the gap and to provide translation techniques for both formalism types, an algorithm translating textual descriptions of automata to automata themselves is necessary.

[**Principal ideas/results**] It was shown in our previous work that discourse context modeling allows to complete information missing from scenarios written in natural language and to translate scenarios to MSCs. The goal of the approach presented in this paper is to translate textual descriptions of automata to automata themselves, by adapting discourse context modeling to texts describing automata.

[**Contribution**] The presented paper shows how the previously developed context modeling approach can be adapted in order to become applicable to texts describing automata. The proposed approach to translation of text to automata was evaluated on a case study, which proved applicability of the approach.

Keywords: requirements analysis, behavior extraction, behavior modeling, natural language processing.

1 Requirements Documents Suffer from Missing Information

At the beginning of every software project, some kind of requirements document is usually written. The majority of these documents are written in natural language, as the survey by Mich et al. shows [5]. This results in the fact that the requirements documents are imprecise, incomplete, and inconsistent, because precision, completeness and consistency are extremely difficult to achieve using mere natural language as the main presentation means. From the linguistic point of view, document authors may introduce three defect types, without perceiving them as defects, cf. Rupp [6]:[1]

[1] The following definitions are translations of the definitions from [6], in German.

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 197–211, 2009.

Deletion: "... is the process of selective focusing of our attention on some dimensions of our experiences while excluding other dimensions. Deletion reduces the world to the extent that we can handle."

Generalization: "... is the process of detachment of the elements of the personal model from the original experience and the transfer of the original exemplary experience to the whole category of objects."

Distortion: "... is the process of reorganization of our sensory experience."

The authors of requirements documents are not always aware of these document defects. Even documents that are precise from the human point of view can omit some facts relevant for behavior specification. The goal of the presented paper is to translate texts to automata despite such defects.

According to Boehm [7], the later an error is found, the more expensive its correction. Thus, it is one of the goals of requirements analysis, to find and to correct the defects of requirements documents. Our previous work [2,3,4] focused on defects in scenarios, specially on the "deletion" defects. The goal of the previous work was to identify missing parts of scenarios written in natural language and to produce Message Sequence Charts (MSCs) containing the reconstructed information. The key idea was to model the discourse context and to infer the missing parts of scenarios from the context. In the case of MSCs, the discourse context model included the set of messages that are sent but not yet answered.

According to our survey of modeling techniques [1], all modeling techniques are either interaction-based (MSC-like) or automata-based. Similarly, texts describing system behavior fall in the same two categories: They either specify scenarios (interaction between system components) or give textual description of automata. The goal of the approach presented in this paper is to translate automata-based textual descriptions to automata. Together with our previous work, this provides extraction of both model types from textual documents. It turns out that, in the case of automata, the discourse context model is simpler than for MSCs and contains only a default initial state for incompletely specified state transitions (see Section 3 for details).

Contribution: The presented paper shows how the idea of discourse context modeling can be transferred to texts describing automata. It shows that a different approach to context modeling, even simpler than the approach developed to translate textual scenarios to MSCs, is sufficient to translate textual descriptions of automata to automata themselves.

Outline: The remainder of the paper is organized as follows: Section 2 introduces the case study used to evaluate the presented approach. Section 3 is the technical core of the paper, it presents and evaluates the approach to translate texts to automata. Sections 4, 5, and 6 present an overview of related work, the summary of the paper, and possible directions for future work, respectively.

2 Case Study: The Steam Boiler

Authors of requirements documents tend either to forget facts that seem obvious to them or they are reluctant to precisely specify the context in which their statements apply. This is quite natural, and is just a part of the human process of focusing attention onto

facts that seem most important at the moment of writing. This results in the problem that even precise specifications, as for example the Steam Boiler Specification [8], used in the presented work, cannot be analyzed on the sentence level.

The Steam Boiler Specification was chosen for the case study, as it was the standard benchmark for several case studies aiming to compare different formalization methods [9]. This specification describes the steam boiler itself and states the requirements to the control program for the steam boiler. The steam boiler system consists of four pumps to provide the steam boiler with water, one controller for every pump, a device to measure the water level in the steam boiler, and a device to measure the quantity of steam coming out of the steam boiler. The goal of the control program is to maintain the water level between predefined marks, in order to prevent damage of the steam boiler. This water level should be maintained even in case of certain equipment failures. In the case of equipment failures, water levels between certain emergency marks are allowed. Water levels above/below emergency marks cause steam boiler damage.

The control program for the steam boiler should support a number of modes: initialization mode, normal mode, degraded mode, rescue mode, and emergency stop mode. For every mode, the specification describes the required program reactions to different operation situations. An example set of rules, applicable in the normal mode, is shown in Table 1. It is easy to see that it makes no sense to analyze every sentence of the specification separately: Some sentences, as for example Sentence 1, Sentence 3, and Sentence 7, do not contain any explicit behavior specification. Others contain behavior information, but cannot be directly translated to state transitions, as they specify the state after the transition only. The initial state, *normal mode*, is common for all transitions and remains unspecified in the sentences describing transitions. Every such omission is a "deletion" defect in the sense of the definitions given in Section 1. The goal of the presented paper is to translate texts to automata despite such defects.

In spite of the fact that separate analysis of every sentence is insufficient even for the relatively well-written Steam Boiler Specification, the existing approaches translat-

Table 1. The steam boiler, specification excerpt (copied from [8])

Normal mode
1. The normal mode is the standard operating mode in which the program tries to maintain the water level in the steam-boiler between N1 and N2 with all physical units operating correctly.
2. As soon as the water level is below N1 or above N2 the level can be adjusted by the program by switching the pumps on or off.
3. The corresponding decision is taken on the basis of the information which has been received from the physical units.
4. As soon as the program recognizes a failure of the water level measuring unit it goes into rescue mode.
5. Failure of any other physical unit puts the program into degraded mode.
6. If the water level is risking to reach one of the limit values M1 or M2 the program enters the mode emergency stop.
7. This risk is evaluated on the basis of a maximal behaviour of the physical units.
8. A transmission failure puts the program into emergency stop mode.

Table 2. Automaton for steam boiler control, manually constructed

Initial mode	Target mode	Transition condition
initialization	initialization	message steam-boiler-waiting not yet received
initialization	emergency stop	unit for detection of the level of steam is defective
initialization	emergency stop	failure of the water level detection unit
initialization	normal	all the physical units operate correctly
initialization	degraded	any physical unit is defective
initialization	emergency stop	transmission failure
normal	rescue	failure of the water level measuring unit
normal	degraded	failure of any other physical unit
normal	emergency stop	the water level is risking to reach one of the limit values
normal	emergency stop	transmission failure
degraded	normal	defective unit repaired
degraded	rescue	failure of the water level measuring unit
degraded	emergency stop	the water level is risking to reach one of the limit values
degraded	emergency stop	transmission failure
rescue	normal	water level measurement unit repaired
rescue	degraded	water level measurement unit repaired
rescue	emergency stop	the unit which measures the outcome of steam has a failure
rescue	emergency stop	the units which control the pumps have a failure
rescue	emergency stop	the water level risks to reach one of the two limit values
rescue	emergency stop	transmission failure

ing textual specifications to models analyze every specification sentence separately (cf. Section 4). It is the goal of the presented work, to capture context information in order to complete information not explicitly mentioned in sentences specifying state transitions.

Table 2 shows the required behavior of the control program, manually constructed on the basis of the specification. This manually constructed automaton will be used to evaluate the proposed text-to-automaton translation procedure in Section 3.

3 Translation of Texts to Automata

The process of text-to-automaton translation is motivated by the already tested and validated algorithm for text-to-MSC translation presented in [2,3,4]. The process of text-to-MSC translation consisted of three steps:

- identification of communicating objects,
- splitting of every sentence into segments,
- for every segment, translation of the segment to an MSC element. An MSC element can be either a message between two communicating objects or an assertion about system state.

The process of text-to-automaton translation follows similar steps: First, the set of potential states is determined. Then, every sentence is split into segments. Finally, segments are translated either to state transitions or to transition conditions. These steps are presented in Sections 3.1-3.3. Section 3.4 presents the results of the evaluation of the presented text-to-automaton translation on the Steam Boiler Specification.

To stay robust, the presented approach uses solely a part-of-speech (POS) tagger [10] on linguistic side and does not use more sophisticated techniques like Discourse Representation Theory (DRT) [11], as the use of techniques like DRT would render the approach highly fragile.

3.1 Identification of States

In our previous work [2,3,4] it was shown that the algorithm for text-to-MSC translation that inspired the presented work is highly sensitive to the proper definition of the set of communicating objects. Thus, it was to expect that the presented algorithm for text-to-automaton translation is sensitive to the proper definition of the set of states.

Identification of states is necessary for later decision whether to translate a particular sentence segment to a state transition. As a first approximation, it is possible to manually extract the names of the states explicitly listed in the text. However, this set of states can be incomplete. In our case study, this incompleteness resulted in missing transitions in the extracted automaton, cf. Section 3.4.

The name of a state can consist of several words, like "emergency stop mode". Furthermore, the same mode can be called, for example, both "emergency stop mode" and "mode emergency stop" in the specification text. To automatically extract the different forms of the mode names, the following procedure was applied:

- The whole text was tagged by a part-of-speech (POS) tagger. The applied tagger [10] has a precision of about 97%, which makes it unlikely to be an error source.
- Following tags were considered: (1) tag "VBD", identifying verbs in the past participle form ("been", "done"), (2) any tag starting with "NN", identifying different noun forms, and (3) tag "JJ", identifying adjectives.
- Following patterns were extracted from the tagged text:
  - Word "mode", followed by any number of substantives (like in "mode|NN rescue|NN"), adjectives (like in "mode|NN normal|JJ"), or verbs in the past participle form (like in "mode|NN degraded|VBD").
  - Any number of substantives, adjectives, or verbs in the past participle form, followed by the word "mode".

  Technically, the extraction of the above patterns from the tagged text was performed by the application of the UNIX tool `grep` with the following regular expressions:
  - `mode|NN ([^|]*|(NN|VBD|JJ))+`
  - `([^ |]*|(NN|VBD|JJ))+ mode|NN`

  Here it is important to emphasize that the signal word "mode" used to identify state names, is specific to the Steam Boiler Specification. For other specification texts, it is necessary to provide other signal words or to use other extraction techniques: For example, in [4] the names of modeling elements were identified as subjects of sentences having particular grammatical features.

The above procedure resulted in the extraction of the word sequences shown in Table 3. This table contains not only the states explicitly defined in the document, but also noise, "standard operating mode". However, as the case study has shown, this noise can be compensated for when constructing the automaton (cf. Section 3.4).

Table 3. Automatically extracted states

"**mode**", followed by other words:	mode emergency stop, mode normal, mode rescue, mode degraded
"**mode**", preceded by other words:	initialization mode, emergency stop mode, normal mode, standard operating mode, rescue mode, degraded mode

The procedure to extract the potential states of the automaton by extracting the named entities with the signal word "mode" was sufficient for the steam boiler case study. In general, it is easy to extend the procedure by adding further signal words. Furthermore, it is possible to integrate the above procedure with grammar-based methods from [4]. Applicability of every particular method depends on the writing style of the concrete document.

3.2 Categories of Sentences

One of the prerequisites for the text-to-automaton translation is the assignment of every (sub)sentence to one of the four categories: "state transition", "transition condition", "context setting", or "irrelevant", cf. Section 3.3. The assignment of sentence segments to categories takes place in the following steps:

1. Splitting of every sentence to segments
2. Assignment of segments to categories on the basis of grammatical information only
3. Re-assignment of segments to categories, by using context information

Each of these steps is described below.

Sentence splitting: To split sentences, just the following assumption is made: punctuation marks are correctly placed to separate subsentences. The splitting process itself is rather simple. Punctuation symbols and the words "if" and "when" are used as splitting marks. Additionally, the conjunctions "and" and "or" are used as splitting marks, unless they directly follow an adjective or a number. This heuristics prevents splitting of expressions like "if the water level lies between N1 and N2, ...". A splitting example is shown in Table 4.

Assignment of segments to categories on the basis of grammatical information: On total, we differentiate four classes of sentence segments:

Table 4. Splitting example

Original sentence
as soon as this signal has been received, the program enters either the mode normal if all the physical units operate correctly or the mode degraded if any physical unit is defective
Splitting
1. as soon as this signal has been received 2. the program enters either the mode normal 3. all the physical units operate correctly 4. the mode degraded 5. any physical unit is defective

- Segments translated to transitions, like “the program enters either the mode normal” in the example in Table 4. Such segments are called “state transition” in the remainder of the paper.
- Segments translated to transition conditions, like “as soon as this signal has been received” in the example in Table 4. Such segments are called “transition condition” in the remainder of the paper.
- Segments that are not translated to any element of the automaton, but setting the context for the subsequent segments, like the first sentence in Table 1. Such segments are called “context setting” in the remainder of the paper.
- Segments that are irrelevant for the text-to-automaton translation, like the third sentence in Table 1. Such segments are called “irrelevant” in the remainder of the paper.

Identification of the four segment classes is possible on the basis of the POS tags and the previously extracted set of states. The identification consists of two phases. In the first phase, every sentence segment is marked on its own. In the second phase, the decision of the first phase is revised by taking the neighbors of the analyzed segment into account. In the first phase, the assignment of the sentence segment to one of the four classes is fairly simple:

- If the sentence segment does not contain any reference to a state (element of the extracted set of states), it is marked as “irrelevant”. This holds, for example, for the first segment in Table 4.
- If the sentence segment contains a reference to a state, but first occurrence of the state is not preceded by a verb, this segment is marked as “context setting”. A word is considered as a verb if the POS tagger assigns a tag starting with “VB” to this word. For example, in Table 1, the header (“normal mode”) and the first sentence set the context for the translation of the following sentences.
- Otherwise, the sentence segment is marked as “state transition”.

Here it is important to emphasize that in the first phase no sentence segment is marked as “transition condition”.

Re-assignment of segments to categories, by using context information: To take context into account, it is necessary to revise the “context setting”-marks first. For example, the fourth segment in Table 4 is marked as “context setting” in the first phase, although it actually specifies a state transition. Here, the following heuristics is applied: If, for a given sentence, any of its segments is marked as “state transition”, then all segments marked as “context setting” are relabeled to “state transition”. This compensates for potentially missing verbs in some sentence segments. In the case of the example shown in Table 4, it marks the fourth segment as “state transition” and leaves the other marks unchanged.

When the marking of segments as “state transition” is finished, it is possible to identify transition conditions:

- If a sentence segment is marked as “irrelevant” and directly precedes a segment marked as “state transition”, then the former segment is relabeled to “transition condition”. This allows to mark the first segment of the example in Table 4, “as soon as this signal has been received”, as “transition condition”.

- After the above step, if a sentence segment is marked as "irrelevant" and directly precedes a segment marked as "transition condition", the former segment is relabeled to "transition condition". This allows to treat compound conditions, like "if message A or message B is received,...".
- If a sentence segment is marked as "irrelevant" and directly follows a segment marked as "state transition", then the former segment is relabeled to "transition condition". This allows to treat conditions like "⟨`some transition`⟩ if ⟨`some condition`⟩".
- After the above step, if a sentence segment is marked as "irrelevant" and directly follows a segment marked as "transition condition", the former segment is relabeled to "transition condition". This allows to treat compound conditions, like "⟨`some transition`⟩ if ⟨`some condition`⟩ or ⟨`some other condition`⟩".

When this relabeling process is finished, we have enough information to translate the text to an automaton.

The process of sentence splitting is purely syntactic, which is its major advantage: this makes sentence splitting independent of writing style of a particular document author. Furthermore, this allows to treat grammatically different types of conditions, like "if something happens" and "as soon as something happens", in a uniform way.

3.3 Context Modeling and Generation of Transitions

When every sentence segment is assigned to one of the four classes ("state transition", "transition condition", "context setting", or "irrelevant"), we can use this information to translate the text to an automaton. The actual text-to-automaton translation exploits the fact that sentence segments marked as "context setting" or "state transition" always refer to a state. The translation algorithm sequentially goes through the marked sentence segments. Depending on the sentence segment class, it performs the following actions:

- Segments marked as "irrelevant" are ignored.
- If the translation algorithm comes across a sentence segment marked as "context setting", the state contained in this segment becomes the default initial state for the transitions generated afterwards.
- If the translation algorithm comes across a sentence segment marked as "state transition", then several transitions are generated. The initial state of the transitions is always the current default initial state (context), the target state is the state taken from the "state transition" segment under analysis. The transitions conditions depend on the neighbors of the segment under analysis:
  - If the "state transition" segment under analysis is followed by a contiguous block of "transition condition" segments, then a state transition is generated for every segment from the "transition condition" block. The textual representation of every "transition condition" segment becomes a transition condition in the generated automaton.
  - If the "state transition" segment under analysis is preceded by a contiguous block of "transition condition" segments, and the "state transition" segment under analysis is the first "state transition" segment of its sentence, then a state transition is generated for every segment of the "transition condition" block, in the same way as above.

- If no state transition can be generated due to the above two rules, the translation algorithm re-analyzes the current "state transition" segment and extracts the word sequence preceding its main verb. The word sequence preceding the main verb becomes the transition condition. This allows to handle constructions like "a transmission failure puts the program into the mode emergency stop". In this case, "a transmission failure puts" becomes the transition condition, cf. Table 5.
- If all the above rules fail, a transition with an empty transition condition is generated.

By inferring the initial states of transitions, the presented algorithm visualizes presuppositions of the document author. This can be used for validation, in particular to proof whether the document author and the document reader interpret the specification in the same way.

The generated automaton is flat: it contains neither parallel nor nested states. Generation of such constructions would require deep semantic analysis, going far beyond capabilities of the existing linguistic tools.

The above rules were implemented in a Java program. This program generates automata represented as table, like Table 2 or Table 5. At the moment, the generated transition conditions are represented in natural language, and are not automatically analyzable. In the long run, the presented approach should be integrated with the approach by Gervasi and Zowghi [12]. Gervasi and Zowghi can translate conditions written in a restricted natural language to logical formulae. This translation would allow to perform further analysis of the automata, like for example completeness of input coverage.

3.4 Evaluation

Three case studies were performed to evaluate the presented approach. The case studies were performed on the same text, namely on Section 4 of the Steam Boiler Specification [8]. This section describes the required behavior of the steam boiler control program. This section was manually cut out of the document and submitted to the text-to-automaton translation.

The case studies differed in the definition of the states:

1. In the first case study, the algorithm for text-to-automaton translation was provided with the set of states explicitly listed in the following specification sentence:
   > The program operates in different modes, namely: initialization, normal, degraded, rescue, emergency stop.

   Thus, the translation algorithm was provided with the following set of states: "initialization mode", "normal mode", "degraded mode", "rescue mode", "emergency stop mode".
2. In the second case study, the algorithm for text-to-automaton translation was provided with the automatically extracted set of states shown in Table 3. The state name "`standard operating mode`" was manually removed from the set, as it does not represent a real state of the control program.
3. In the third case study, the algorithm for text-to-automaton translation was again provided with the automatically extracted set of states shown in Table 3. In contrast to the second case study, however, "`standard operating mode`" was not removed from the set.

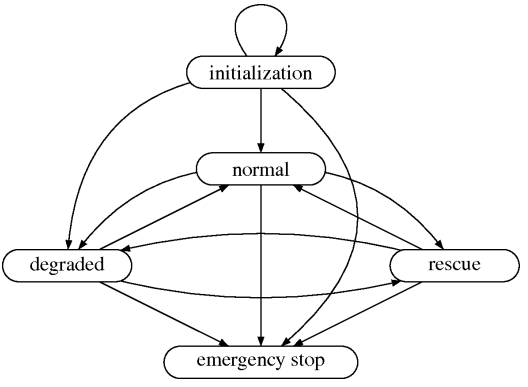
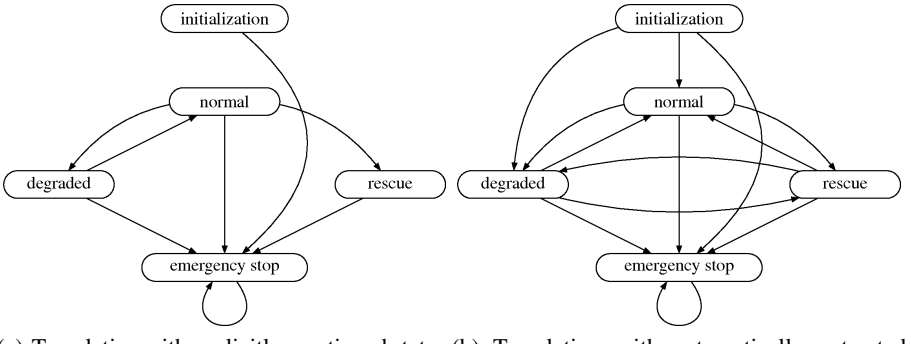


Fig. 1. Automaton for steam boiler control, manually constructed



(a) Translation with explicitly mentioned states (b) Translation with automatically extracted states, cf. Section 3.1

Fig. 2. Automaton for steam boiler control, automatically extracted

The first case study produced the automaton shown in Figure 2(a). When compared with the manually constructed automaton, shown in Figure 1, the automaton in Figure 2(a) definitely lacks several state transitions.

The second and the third case studies produced the same automaton, shown in Figure 2(b) and Table 5. Interestingly, the state "`standard operating mode`" did not result in any additional state transitions in the third case study. As the output algorithm ignores such standalone states, this state is not presented in Figure 2(b).

To evaluate the text-to-automaton translation, we compare the manually constructed automaton with the generated one. If we compare the graphical representations, i.e. Figure 1 with Figure 2(b), we see that the automata coincide, except for the loop in the "`initialization`" mode in Figure 1 and the loop in the "`emergency stop`" mode in Figure 2(b). Manual analysis of the steam boiler specification shows that the behavior in the emergency stop mode is underspecified. It can be interpreted both as a loop and as its absence: "once the program has reached the emergency stop mode, the physical environment is then responsible to take approrpiate actions, and the

Table 5. Automaton for steam boiler control, automatically extracted

Initial mode	Target mode	Transition condition
initialization	emergency stop	the unit for detection of the level of steam is defective – that is, when v is not equal to zero – the program enters
initialization	emergency stop	the program realizes a failure of the water level detection unit it enters
initialization	normal	all the physical units operate correctly
initialization	degraded	any physical unit is defective.
initialization	emergency stop	a transmission failure puts
normal	rescue	as soon as the program recognizes a failure of the water level measuring unit it goes
normal	degraded	failure of any other physical unit puts
normal	emergency stop	the water level is risking to reach one of the limit values m1 or m2 the program enters
normal	emergency stop	a transmission failure puts
degraded	normal	once all the units which were defective have been repaired, the program comes
degraded	rescue	as soon as the program sees that the water level measuring unit has a failure, the program goes
degraded	emergency stop	the water level is risking to reach one of the limit values m1 or m2 the program enters
degraded	emergency stop	a transmission failure puts
rescue	degraded	as soon as the water measuring unit is repaired
rescue	normal	
rescue	emergency stop	it realizes that one of the following cases holds: the unit which measures the outcome of steam has a failure,
rescue	emergency stop	the units which control the pumps have a failure,
rescue	emergency stop	the water level risks to reach one of the two limit values.
rescue	emergency stop	a transmission failure puts
emergency stop	emergency stop	the program stops.

program stops". As for the loop in the "`initialization`" mode, its extraction requires semantic analysis, going beyond the capabilities of the available linguistic tools. This loop stems from the sentence "the program enters a state in which it waits for the message steam-boiler-waiting to come from the physical units". It is not yet possible for linguistic tools to interpret the word "wait" as a state loop. Hard-coding generation of loops for words like "wait" would make the approach highly dependent on the writing style and would make generalization extremely difficult.

If we compare the table representations, Table 2 and Table 5, we see that they coincide except for the already discussed loops, if we ignore phrasings for the transition conditions. Furthermore, due to the applied sentence splitting algorithm, transition conditions in Table 5 are sometimes grammatically incomplete. A closer analysis shows that the transition conditions in Table 2 and Table 5 are semantically equivalent and differ in their lexical representation only. The only exception is the transition from the rescue mode to the normal mode that lacks a transition condition. This transition originates from the sentence "as soon as the water measuring unit is repaired, the program returns into mode degraded or into mode normal". The exact transition condition to "`mode normal`" is not specified. In Table 2 it was just guessed that the transition conditions to "`mode normal`" and "`mode degraded`" coincide.

Anyway, none of the automata, neither the manually constructed nor the automatically extracted can be directly used for further system development. Both automata rather serve to visualize the specification and thus to ease its validation. For this reason differences in lexical representations of transition conditions are unimportant.

To summarize, the presented approach to text-to-automaton translation is able to translate texts about automata to automata themselves and the translation result is precise enough to be used for behavior validation.

4 Related Work

Ryan [13] claimed that natural language processing is not mature enough to fully automate requirements engineering. In the same paper he admitted, however, that natural language processing can be useful to support human analysts. There was a lot of work aiming to support human analysts in recent years.

There are three areas where natural language processing is applied to requirements engineering: assessment of document quality, identification and classification of application specific concepts, and analysis of system behavior. Approaches to the assessment of document quality were introduced, for example, by Rupp [6], Fabbrini et al. [14], Kamsties et al. [15], and Chantree et al. [16]. These approaches have in common that they define writing guidelines and measure document quality by measuring the degree to which the document satisfies the guidelines. These approaches have a different focus from the approach presented in this paper: their aim is to detect poor phrasing and to improve it, they do not target at behavior analysis.

Another class of approaches, like for example those by Goldin and Berry [17], Abbott [18], or Sawyer et al. [19] analyzes the requirements documents, extracts application specific concepts, and provides an initial model of the application domain. These approaches do not perform any behavior analysis, either.

The approaches analyzing system behavior, as for example those by Vadera and Meziane [20], Gervasi and Zowghi [12], and Avrunin et al. [21] translate requirements documents to executable models by analyzing linguistic patterns. In this sense they are similar to the approach presented in this paper. Vadera and Meziane propose a procedure to translate certain linguistic patterns into first order logic and then to the specification language VDM, but they do not provide automation for this procedure. Gervasi and Zowghi go further and introduce a restricted language, a subset of English. They automatically translate textual requirements written in this restricted language to first order logic. The approach by Avrunin et al. is similar to the approach by Gervasi and Zowghi in the sense that it introduces a restricted natural language. The difference lies in the formal representation means: Gervasi and Zowghi stick to first order logic, Avrunin et al. translate natural language to temporal logic. The presented paper goes further than the above two approaches, as the language is not restricted, and the assumptions about phrasing are minimal: It is solely assumed that punctuation marks are correctly placed.

To summarize, to the best of our knowledge, there is no approach to requirements documents analysis, that is able to analyze documents written in non-restricted language, model context information and use this context modeling to complete the information missing from the text when translating the text to an executable model.

5 Summary

The approach presented in this paper automates parts of the step from requirements documents to design. Despite minimal assumptions about the structure of the sentences to be translated, the approach is effective, which was shown in case studies. The translation of texts to design imitates the way how human analysts would model the discourse context. This context model is then applied to infer information not explicitly stated in the behavior specification.

The presented approach relies, in its pure form, on the writing style of the Steam Boiler Specification. Under following assumptions, it can be generalized and applied to other specifications too:

The set of system states is known: In the presented work, the set of states was extracted from the specification, but, in general, it is possible to provide the approach with a predefined set of states. It is important that the provided set of states be complete: if a state is missing, some sentences may be wrongfully identified as "irrelevant" instead of "context setting" or "state transition", which would definitely hurt the correctness of the generated automaton. Presence of noise states ("standard operating mode" in the presented case study), however, can be compensated for, as long as the noise states do not occur in sentences identified as "state transition".

Sentences describing state transitions contain a reference to the target state, as in "if ..., the system goes into ⟨target state⟩". Given that the initial state of a state transition can be inferred from the context, this allows to extract a complete state transition.

Context setting is stated explicitly, either in paragraph titles or in describing sentences like "⟨context state⟩ is the state in which ..."

Comma setting is correct: "if ⟨condition⟩, then ⟨action⟩" or "⟨action⟩ if ⟨condition⟩".

Before being used in the further development process, the generated behavior model has to be validated. Validation is necessary for at least two reasons:

- The original requirements document can contain inconsistencies or omissions.
- The applied linguistic tools do not offer 100% precision, and errors introduced by the linguistic tools may interfere with the presented heuristic for automata construction.

Validation of the produced automaton can make apparent the ambiguities or omissions in the document, not perceived by a human analyst. Validation of the automaton becomes especially valuable if the automaton generated by the presented approach radically differs from the manually constructed automaton. This can mean that the requirements text has several interpretations and thus should be made more precise before used in the further development steps. When the generated automaton is validated, it can be used in the further development process. Thus, the presented approach makes a contribution both to document improvement and validation and to the transition from requirements to design.

6 Future Work

The approach presented in this paper is a proof-of-concept that discourse context modeling can be successfully applied to translate specification texts to behavior models. It can be further developed in different directions. First of all, the case study used to evaluate the approach was relatively small. A larger case study would allow more significant conclusion about the precision of the proposed approach. Secondly, the generated transition conditions, as for example those shown in Table 5, sometimes contain unnecessary words. This problem arises from the fact that it is not possible to determine the boundaries of the subordinate clauses without parsing the sentence. In the presented work, mere part-of-speech (POS) tagging was applied instead of parsing, as POS tagging is much more precise (97% precision for tagging [10] vs. approx. 80% for parsing [22]). To combine the advantages of both technologies, the presented approach can be augmented in such a way that POS tagging is used for the actual text-to-automaton translation and parsing is used to determine clause boundaries. In this way it is possible to generate better transition conditions.

To validate the generated automata, a technique similar to CREWS-SAVRE [23] can be applied: In the original version of CREWS-SAVRE, a sequence of events is taken as input, and, for this sequence, questions like "What happens if the specified event does not occur?" are generated. In a similar way, for every state transition of the generated automaton, we could generate questions like "What happens if the input signal necessary for the transition does not occur?", "What happens if the input signal necessary for the transition occurs several times?", etc.

Developments sketched above would further improve the presented approach and make it industrially applicable.

References

1. Kof, L., Schätz, B.: Combining aspects of reactive systems. In: Broy, M., Zamulin, A.V. (eds.) PSI 2003. LNCS, vol. 2890, pp. 344–349. Springer, Heidelberg (2004)
2. Kof, L.: Scenarios: Identifying missing objects and actions by means of computational linguistics. In: 15th IEEE International Requirements Engineering Conference, New Delhi, India, pp. 121–130. IEEE Computer Society Conference Publishing Services, Los Alamitos (2007)
3. Kof, L.: Treatment of Passive Voice and Conjunctions in Use Case Documents. In: Kedad, Z., Lammari, N., Métais, E., Meziane, F., Rezgui, Y. (eds.) NLDB 2007. LNCS, vol. 4592, pp. 181–192. Springer, Heidelberg (2007)
4. Kof, L.: From Textual Scenarios to Message Sequence Charts: Inclusion of Condition Generation and Actor Extraction. In: 16th IEEE International Requirements Engineering Conference, Barcelona, Spain, pp. 331–332. IEEE Computer Society, Los Alamitos (2008)
5. Mich, L., Franch, M., Novi Inverardi, P.: Market research on requirements analysis using linguistic tools. Requirements Engineering 9, 40–56 (2004)
6. Rupp, C.: Requirements-Engineering und -Management. In: Professionelle, iterative Anforderungsanalyse für die Praxis, 2nd edn., Hanser–Verlag (2002) ISBN 3-446-21960-9
7. Boehm, B.W.: Software Engineering Economics. Prentice-Hall, Englewood Cliffs (1981)
8. Abrial, J.R., Börger, E., Langmaack, H.: The steam boiler case study: Competition of formal program specification and development methods. In: Abrial, J.R., Borger, E., Langmaack, H. (eds.) Dagstuhl Seminar 1995. LNCS, vol. 1165. Springer, Heidelberg (1996)

9. Abrial, J.-R., Börger, E., Langmaack, H. (eds.): Dagstuhl Seminar 1995. LNCS, vol. 1165. Springer, Heidelberg (1996)
10. Curran, J.R., Clark, S., Vadas, D.: Multi-tagging for lexicalized-grammar parsing. In: 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Sydney, Australia, July 17-21 (2006)
11. Blackburn, P., Bos, J., Kohlhase, M., de Nivelle, H.: Inference and computational semantics. CLAUS-Report 106, Universität des Saarlandes, Saarbrücken (1998)
12. Gervasi, V., Zowghi, D.: Reasoning about inconsistencies in natural language requirements. ACM Trans. Softw. Eng. Methodol. 14, 277–330 (2005)
13. Ryan, K.: The role of natural language in requirements engineering. In: Proceedings of IEEE International Symposium on Requirements Engineering, pp. 240–242. IEEE Computer Society Press, Los Alamitos (1992)
14. Fabbrini, F., Fusani, M., Gnesi, S., Lami, G.: The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool. In: 26th Annual NASA Goddard Software Engineering Workshop, Greenbelt, Maryland, pp. 97–105. IEEE Computer Society, Los Alamitos (2001)
15. Kamsties, E., Berry, D.M., Paech, B.: Detecting ambiguities in requirements documents using inspections. In: Workshop on Inspections in Software Engineering, Paris, France, pp. 68–80 (2001)
16. Chantree, F., Nuseibeh, B., de Roeck, A., Willis, A.: Identifying nocuous ambiguities in natural language requirements. In: RE 2006: Proceedings of the 14th IEEE International Requirements Engineering Conference (RE 2006), Washington, DC, USA, pp. 56–65. IEEE Computer Society Press, Los Alamitos (2006)
17. Goldin, L., Berry, D.M.: AbstFinder, a prototype natural language text abstraction finder for use in requirements elicitation. Automated Software Eng. 4, 375–412 (1997)
18. Abbott, R.J.: Program design by informal English descriptions. Communications of the ACM 26, 882–894 (1983)
19. Sawyer, P., Rayson, P., Cosh, K.: Shallow knowledge as an aid to deep understanding in early phase requirements engineering. IEEE Trans. Softw. Eng. 31, 969–981 (2005)
20. Vadera, S., Meziane, F.: From English to formal specifications. The Computer Journal 37, 753–763 (1994)
21. Smith, R.L., Avrunin, G.S., Clarke, L.A., Osterweil, L.J.: Propel: an approach supporting property elucidation. In: ICSE 2002: Proceedings of the 24th International Conference on Software Engineering, pp. 11–21. ACM Press, New York (2002)
22. Clark, S., Curran, J.R.: Wide-coverage efficient statistical parsing with ccg and log-linear models. Comput. Linguist. 33, 493–552 (2007)
23. Maiden, N.A.M.: CREWS-SAVRE: Scenarios for Acquiring and Validating Requirements. Automated Software Engineering 5, 419–446 (1998)

A Requirements Reference Model for Model-Based Requirements Engineering in the Automotive Domain

Birgit Penzenstadler[1], Ernst Sikora[2], and Klaus Pohl[2]

[1] Software & Systems Engineering, Technische Universität München, Germany
penzenst@in.tum.de
[2] Software Systems Engineering, Universität Duisburg-Essen, Germany
{ernst.sikora,klaus.pohl}@sse.uni-due.de

Abstract. [**Context and motivation**] The use of conceptual models in automotive requirements engineering is impaired due to the lack of appropriate modelling guidelines. [**Question/problem**] The goal of this paper is to propose a requirements reference model that serves as the basis for defining such guidelines. [**Principal ideas/results**] The reference model distinguishes three abstraction layers and three content categories for requirements models. [**Contribution**] The reference model has been successfully applied in the REMsES project to support the development of a model-based requirements engineering approach for the automotive domain.

Keywords: requirements models, reference models, abstraction layers.

1 Motivation

Requirements for automotive embedded systems are documented predominantly using natural language. However, natural language offers limited support for dealing with large and complex requirements specifications. Model-based requirements engineering (RE) can help to overcome some of the difficulties related to natural language requirements (see e.g. [1]). In model-based RE, conceptual models such as goal models, scenario models, and function models are used to specify requirements. However, according to our experience, the lack of a comprehensive guidance impairs the use of models in requirements engineering practice.

We have observed that requirements engineers in the automotive domain find it difficult to create requirements models that support them in managing the high system complexity. In particular, when creating requirements models, the problem arises which is the right level of abstraction for the model. Quite often, the models that we have found in practice were at a very detailed technical level and therefore insufficient for managing a high system complexity (cf. [2]). Furthermore, in many cases, requirements engineers are not sufficiently familiar with the advantages and disadvantages of different types of requirements models. Hence they have difficulties, e.g. in finding the specification technique that satisfies their needs best.

The main goal of our work is to provide requirements engineers in the automotive domain with a comprehensive guidance for the creation of models in requirements engineering. The requirements reference model presented in this paper forms the core

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 212–217, 2009.

of this guidance. The reference model is based on two structuring principles: the distinction between three abstraction layers at which requirements models can be developed and three essential categories of content to be documented by means of models. The model supports documenting requirements for the system as a whole as well as for individual system components. Furthermore, the reference model suggests six specification techniques for documenting the different kinds of content. It has been developed in collaboration with our partners from the automotive industry. The model has been applied for deriving detailed modelling guidelines for the different abstraction layers, content categories, and specification techniques. In addition, it has been applied in an approach for supporting the intertwined development of requirements and architecture (see [3]).

In this paper, we motivate and explain the key principles of our requirements reference model. The detailed guidelines derived from the reference model are not reported in this paper. The paper is organised as follows. Section 2 presents the key concepts of the requirements reference model. Section 3 presents the most relevant related work. Section 4 concludes the paper and provides an outlook on future work.

2 The Requirements Reference Model

In this section, we explain the key structuring principles of our requirements reference model (see Fig. 1): the three abstraction layers and the three content categories. In addition, we briefly discuss the relationship between the specification techniques included in the reference model and the structuring principles of the model.

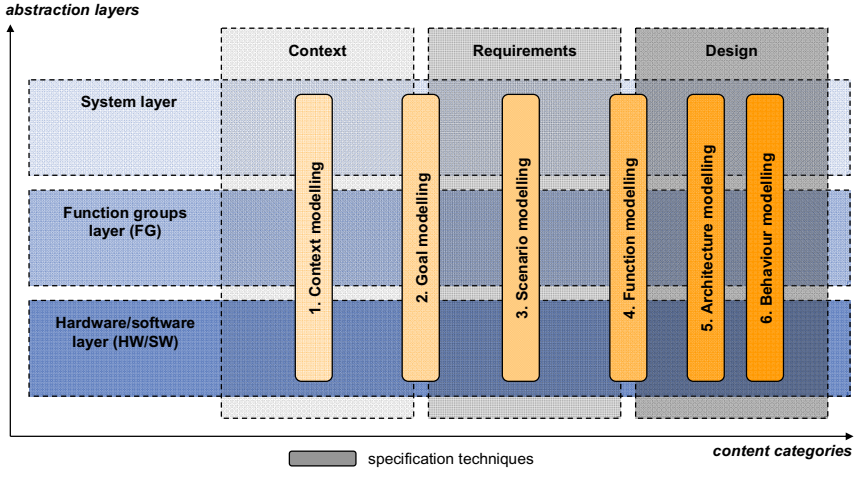


Fig. 1. Structure of the requirements reference model

2.1 Abstraction Layers

Requirements engineers in the embedded systems domain (e.g. in the automotive domain) are faced with requirements at different levels of abstraction (see e.g. [4]).

The requirements range from high-level system requirements to detailed technical requirements for individual software and hardware components. This diversity of requirements demands a systematic way of defining how to deal with each requirement according to its granularity and abstraction level. A well-known solution to deal with varying levels of abstraction of requirements is to establish multiple layers of abstraction and assign each requirement to the appropriate abstraction layer (see e.g. [4]). In the literature and in practice, different hierarchies of abstraction layers are defined and used (see Section 3). In the REMsES project, we have adopted a hierarchy of three abstraction layers: system layer, function groups layer, and HW/SW layer (see Fig. 1). These layers can be characterised as follows:

- **System layer:** At the system layer, the stakeholders take an "outside" (or a black box) view of the system. Requirements models at this layer focus on the usage of the system by its human users and other systems. Since these models are essentially free of technical details, they have a significantly lower complexity.
- **Function groups layer:** At the function groups layer, the system is viewed as a network of interacting, logical units. The system structure at this layer is obtained by a logical decomposition of the system into units of coherent functionality. We refer to these units as "function groups".
- **Hardware/software layer:** At the hardware/software layer, a coarse-grained partitioning of the system functionality into HW and SW is defined. For this purpose, the system is decomposed into (coarse-grained) HW and SW components.

Requirements models that are defined at the system layer have a significantly lower complexity than the models at the function groups layer and the HW/SW layer. The function groups layer allows requirements engineers to define requirements that are more detailed than the requirements defined at the system layer. At the same time, it allows to postpone defining some details about the technical realisation of the requirements. At the HW/SW layer, the requirements for the individual HW and SW components of the system are defined. These requirements refine the requirements that are defined at the function groups layer. However, the decomposition of the system into HW and SW components is rather coarse-grained, i.e. the detailed HW and SW design is not in the scope of the HW/SW layer. In addition, the HW components considered at the HW/SW layer are essentially the peripheral devices of the system such as sensors and actuators needed to realise the interactions of the system with its environment.

2.2 Content Categories

By analysing requirements documents in the automotive domain, we have identified three main categories of content to be documented by means of conceptual models: context, requirements, and (coarse-grained) design. The content categories context and design have been included in our reference model as they contain essential information for the requirements engineering process. Furthermore the inclusion of the three categories in our reference model shall increase the requirements engineers' awareness for the need to distinguish between these three categories. This shall avoid the intermingling of context information, requirements, and design during modelling. The context category and the design category can be characterised as follows:

- **Context:** We distinguish between the business context, the stakeholder context, and the operational context of an automotive embedded system (see [5]). The operational context of an automotive system comprises, for instance, actors and environmental variables. Actors are persons, systems, or devices that interact with the system. Environmental variables are the variables that the system must monitor or control by means of sensors and actuators. In order to specify the requirements for the system, the requirements engineers need detailed knowledge about the context. We suggest to use conceptual models to provide an overview of the relevant aspects of the context as well as to support the detailed analysis of specific context aspects (e.g. protocols that the system must adhere to).
- **Design:** A certain amount of design information in the requirements document of an embedded system is inevitable, because knowledge about the major system components is required for refining high-level requirements into detailed, technical requirements (see [3]). In addition, it is quite common in the automotive domain to include design artefacts in the requirements document in order to hint at the intended solution or to outline a feasible solution, respectively. We have included the design category in our reference model to allow for capturing the major parts of the system and the essential relationships among these parts. Furthermore, by defining design as a content category of its own, we aim at supporting developers in documenting requirements and design in separate models rather than (unknowingly) including design information in requirements models. Thereby, the complexity of the requirements models is further reduced.

2.3 Specification Techniques

Our artefact reference model includes six specification techniques that the requirements engineers of an automotive system can use for documenting the three different kinds of content at the three abstraction layers (Fig. 1). The descriptions of the specification techniques are included in the documentation of our requirements reference model. The techniques have been selected based on the following constraints: (1) At least one specification technique is needed for each content category. (2) The reference model should support goal- and scenario-based requirements engineering. (3) The reference model should support the reasoning about system functions since functions are an important means for specifying and reasoning about automotive system requirements. (4) The reference model should allow for a detailed specification of components (e.g. in terms of the component interfaces and the behaviour of the components at their interfaces).

As shown in Fig. 1, some specification techniques are assigned to a single category while other techniques are assigned to two categories. For instance, scenario modelling is assigned to the requirements category, since scenario models are suited particularly well for documenting the required interactions of a system or a component. Goal modelling is assigned to two categories, because goal modelling approaches support the documentation of both, information or assumptions about the context as well as functional and quality requirements equally well.

The abstraction layers do not affect the choice of the specification technique but still have a significant influence on model creation and the resulting models. For instance, a scenario at the system layer documents the interactions among the system

and its actors whereas a scenario at the HW/SW layer documents, e.g. the interactions of a software component with other components such as sensors and actuators.

When defining detailed modelling guidelines, one must choose a specific coordinate with respect to the dimensions “abstraction layer”, “content category”, and “specification technique”. In the REMsES project, detailed guidelines for creating models at each possible coordinate have been defined. A simple example of a guideline for the coordinates <system layer, requirements, goal modelling> is “Define goals at the system layer independently of a specific decomposition of the system at the lower abstraction layers.”

3 Related Work

The approaches presented in the literature that come closest to our reference model are the Requirements Abstraction Model (RAM) and the Zachman Framework.

The RAM [4] defines a hierarchy of five abstraction layers for requirements: product level, feature level, function level, and component level. At the product level requirements with an abstraction level similar to product strategies are defined. The feature level captures requirements which describe characteristic properties of the system. The function level describes individual user functions. The component level captures requirements which are close to exemplary solutions. In principle, a rough mapping between the layers of the RAM and our reference model can be established. However, the RAM does not provide an explicit support for the two kinds of system decomposition supported by our reference model: the decomposition of the system into functional components and the decomposition into HW/SW components. Secondly, specification techniques are not considered in the RAM since the RAM does not address model-based requirements engineering. Thirdly, since the RAM does not distinguish between the three basic content categories context, requirements, and design, it is not suited for guiding the creation of models which clearly distinguish between these three kinds of content.

In the Zachman Framework [6], a hierarchy of five layers is defined to represent the viewpoints of the different stakeholders of an information system. For each layer, the framework distinguishes between six information categories. However, although both, the Zachman Framework and our reference model have a two-dimensional structure, the definitions of the layers and categories are conceptually different. The Zachman Framework neither considers the refinement of requirements based on the decomposition of the system nor the distinction between context, requirements, and design. Furthermore, the layers and categories of the Zachman Framework are difficult to adapt to the needs of the automotive domain.

4 Conclusion and Outlook

In this paper, we have presented the key structuring principles of our requirements reference model. The reference model has been developed in collaboration with industrial partners from the automotive domain. The model considers the needs of this domain by means of a hierarchy of three abstraction layers and three content

categories. It provides six specification techniques for documenting the different kinds of content at the three abstraction layers.

The reference model has been applied in the REMsES project for defining a comprehensive set of modelling guidelines. An initial, experimental evaluation of these guidelines indicated a better quality of the resulting requirements artefacts compared to a state-of-practice process that was based on the Volere approach (see [7]). The results of the experiment are presented in [8]. In addition, the reference model is used in the COSMOD-RE method for supporting the intertwined development of requirements and architecture for embedded systems [3]. The successful application of the requirements reference model in the REMsES project and in COSMOD-RE indicates the utility of this reference model.

Our ongoing work focuses on the further enhancement of the modelling guidelines that were defined in the REMsES project. In addition we aim at strengthening the evidence of the utility of these guidelines and the underlying reference model for supporting the development of high-quality requirements models. Furthermore, we seek to transfer and adapt our reference model to other domains such as avionic systems, automation systems, or medical systems.

Acknowledgements. This paper was partly funded by the BMBF project REMsES, grant no. 01 IS F06 D.

References

1. Pretschner, A., Broy, M., Kruger, I.H., Stauner, T.: Software Engineering for Automotive Systems: A Roadmap. Future of Software Engineering, 55–71 (2007)
2. Weber, M., Weisbrod, J.: Requirements Engineering in Automotive Development: Experiences and Challenges. IEEE Software 20(1), 16–24 (2003)
3. Pohl, K., Sikora, E.: COSMOD-RE: Supporting the Co-design of Requirements and Architectural Artifacts. In: Proc. of the 15th IEEE Intl. Conf. on Requirements Engineering (RE 2007), pp. 258–261 (2007)
4. Gorschek, T., Wohlin, C.: Requirements Abstraction Model. Requirements Engineering 11(1), 79–101 (2006)
5. Weyer, T., Pohl, K.: Eine Referenzstrukturierung zur modellbasierten Kontextanalyse im Requirements Engineering softwareintensiver eingebetteter Systeme. In: Modellierung. LNI, vol. 127, pp. 181–196 (2008)
6. Sowa, J.F., Zachman, J.A.: Extending and Formalising A Framework for Information Systems Architecture. IBM Systems Journal 31(3), 590–616 (1992)
7. Robertson, J., Robertson, S.: Volere Standard Specification (2006), `http://volere.co.uk/`
8. Leuser, J., Porta, N., Bolz, A., Rachke, A.: Empirical Validation of a Requirements Engineering Process Guide. In: 13th Intl. Conf. on Evaluation and Assessment in Software Engineering (EASE 2009) (to appear, 2009)

Quality Requirements in Practice: An Interview Study in Requirements Engineering for Embedded Systems

Richard Berntsson Svensson[1], Tony Gorschek[2], and Björn Regnell[1]

[1] Lund University, Department of Computer Science, PO Box 118,
221 00 Lund, Sweden
{Richard.Berntsson_Svensson,Bjorn.Regnell}@cs.lth.se
[2] Blekinge Institute of Technology, School of Engineering, PO Box 520,
372 25 Ronneby, Sweden
tony.gorschek@bth.se

Abstract. [Context and motivation] In market-driven software development it is crucial, but challenging, to find the right balance among competing quality requirements (QR). **[Problem]** In order to identify the unique challenges associated with the selection, trade-off, and management of quality requirements an interview study is performed. **[Results]** This paper describes how QR are handled in practice. Data is collected through interviews with five product managers and five project leaders from five software companies. **[Contribution]** The contribution of this study is threefold: Firstly, it includes an examination of the interdependencies among quality requirements perceived as most important by the practitioners. Secondly, it compares the perceptions and priorities of quality requirements by product management and project management respectively. Thirdly, it characterizes the selection and management of quality requirements in down-stream development activities.

Keywords: Quality requirements; Non-functional requirements; Requirements engineering; Market-driven requirements engineering; Empirical study.

1 Introduction

The complexity of software systems is determined by both functionality and by quality aspects such as performance, reliability, accuracy, security, and usability [6]. These quality aspects, or non-functional requirements are subsequently called quality requirements (QR). It is commonly acknowledged that the handling and balance of QR are an important and difficult part of the requirements engineering (RE) process [16], playing a critical role in software development [6]. However, the situation is even more complex in a market-driven development situation [1]. In market-driven development, the flow of requirements is not limited to one project, and the requirements are generated from internal (e.g., engineers) and external (e.g., customers) sources [15]. Also, to achieve high-quality in complex embedded systems, a combination of experience and knowledge from different disciplines is needed [19]. This may lead to

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 218–232, 2009.

communication difficulties and difficulties in achieving the required quality level [19]. QR often specify certain quality levels and QR are in many cases possible to quantify [22]. Quantification is important, not only for understanding QR [16], but also for planning [24]. Not dealing, or ineffectively dealing with QR may lead to more expensive software and longer-time-to-market [10], or in worst case, failures in software development [2, 11]. Studies [3, 9] have showed cases where QR are the most expensive and difficult aspects to handle, and according to Chung et al., QR are often poorly understood in comparison to less critical aspects of software development [6]. To be able to improve how QR are handled it is important to understand their characteristics [22], how they are used and prioritized in industry, as well as the challenges of dealing with QR. This paper presents an empirical study performed in industry to investigate these aspects as well as complement other RE surveys as few of them have focused on the specific challenges related to QR.

Two main perspectives on QR are studied in this paper [14]. First, the product perspective. Product managers are responsible for the overall product perspective and the selection of the overall planning of the product evolution and offering are elicited (for further elaboration see [26]). Second, the project perspective is studied through the project leader, responsible for managing and prioritizing within the realization phases. The two perspectives are also compared, studying the alignment between project and product managers. The purpose of this study is to discover and describe how QR are handled in practice, both from the product manager and the project leader's perspective, which is important since communication problems are a challenge in market-driven software development [18]. In addition, the effects of not dealing, or ineffectively dealing with QR are also investigated. The paper presents the results of an empirical study that includes data collected from ten practitioners (five product managers and five project leaders) at five companies in Sweden.

The reminder of this paper is organized as follows. In section 2, the background and related work are presented. The research methodology is described in Section 3, while Section 4 presents the results and relates the findings to previous studies. Section 5 gives a summary of the main conclusions.

2 Background and Related Work

There are several surveys that concern or include RE related challenges. Curtis et al. reported the first significant field survey of practices [8]. Even though the study does not have a focus on RE, challenges related to RE were identified, including communication breakdowns and conflicting requirements. Next, a study by Chatzoglou identifies problems with the RE process, the challenges presented are e.g., lack of resources and poor quality of tools and techniques in the RE process [5].

Lubars et al. published a field study on requirements modeling [21]. The presented challenges include vaguely stated requirements and difficulties with prioritization of requirements. In addition, Lubars at el. identified challenges in relation to specification of performance requirements (a type of QR) such as the rationale is not always obvious and difficulties to associate performance requirements with parts of dataflow or control flow specifications. In addition, a field study by Kamsties et al. includes small and medium sized enterprises [17]. The identified challenges include implementation of

new requirements may cause unpredictable interaction with existing requirements, requirements are not traceable, and that requirements are too vague to test. Kamsties et al. also indentified a challenge related to specification of graphical user interfaces (usability requirements, a type of QR). Furthermore, Karlsson et al. published a study with solely focus on challenges in market-driven software development [18]. The presented challenges include communication problems between marketing and development, and requirements prioritization. Karlsson et al. also indentified challenges in relation to QR. One challenge is related to QR interdependencies, which was identified as a major problem. Quality requirements can influence a large part of the functionality or other QR. This is not only related to finding the existing interdependencies, but also assessing to what extent that requirements affect each other, and determining how to deal with this. In addition, problems with considering quality requirements in release planning were identified.

Several studies [4, 6, 7] have looked at requirements interdependencies; for example, Carlshamre et al. identified six different interdependency types in industry [4]. Research related to classification and measurement of QR are also introduced in literature [16, 22]. Olsson et al. conclude that for a method to be successful, it is important that it is flexible enough to handle the diverse nature of QR [22].

The focus of the above mentioned studies have not been primarily on QR, but QR-related findings emerged as parts of the results. This paper presents a study with the primary focus on QR and how they are managed in the RE process.

3 Research Method

The study was carried out using a qualitative research approach [25]. Qualitative research aims to investigate and understand phenomena within its real life context. A qualitative research approach is useful when the purpose is to explore an area of interest, and when the aim is to improve the understanding of phenomena. The purpose of this study is to gain in-depth understanding of QR within market-driven embedded systems companies. The following research questions (see Table 1) provided a focus for our empirical investigation.

It is important to understand an organizations alignment in terms of QR, otherwise there may be a mismatch between product management [26] and project leadership. Project leaders may down prioritize quality aspects that are considered important by product managers and vice verca. In addition, interdependencies are important to understand since QR may influence a large part of the system [18]. Kamsties et al. found that requirements are often too vague to test [17], therefore, it is important to investigate if QR are quantified in industry. Also, dismissal of QR from projects may have an impact on the predicted return of investment, as well as the cost for the customers. Finally, QR are a difficult part of the RE process [16], however; not all challenges in relation to QR may be of major concern for industry. Therefore, it is important to understand what challenges are critical and which ones are adequately handled today.

Table 1. Research questions

Research Questions (QR = Quality Requirements)
RQ1: Is there any difference in the views of what quality requirements are the most important between product managers and project leaders?
RQ2: What interdependencies between QR are present in the companies?
RQ2.1: What types of interdependencies are deemed most important by practitioners, and is there any difference between the view of product managers and project leaders in this regard?
RQ2.2: To what extent are interdependencies elicited, analyzed and documented in industrial practice?
RQ3: Are QR specified in a measurable manner?
RQ4: To what extent are QR dismissed from projects after project initiation?
RQ4.1: If QR are dismissed, is any consequence analysis performed?
RQ5: What QR challenges are articulated as critical by the practitioners themselves?
RQ6: What QR aspects do the companies feel confident as being adequately handled today?

3.1 Research Design and Data Collection

The study uses semi-structured interviews enabling exploratory discussion between the researcher and the interviewee. The study was conducted in two stages: first the data from each company was collected and analyzed. Secondly, the combined data from all participating companies was collected and analyzed. The criteria for selecting companies were based on our corporate contacts within industry. Five market-driven software companies participate. From each company, one product manager (PM) and one project leader (PL) from the same project were interviewed, resulting in ten data points. The study consists of three phases: planning, data collection, and analysis.

Planning: The first phase of the study involved a brainstorming and planning meetings to design the study and to identify different areas of interests. A combination of maximum variation sampling and convenience sampling was used to select companies within our industrial collaboration network [23]. The included companies vary in respect to size, type of product, and application domain, a rudimentary characterization can be see in Table 2 (more details are not revealed for confidentiality reasons). The interview instrument was designed with respect to the different areas of interest and inspiration from [18]. To test the interview instrument[1], two pilot interviews were conducted prior to the industry study.

Data collection: The study used a semi-structured interview strategy [25]. All interviews were attended by one interviewee and one interviewer. First, the purpose of the study and a general explanation of QR were presented and then questions about the different areas of interests in relation to QR were discussed in detail. All interviews varied between 40 and 90 minutes.

Analysis: The content analysis [25] involved creating categories where interesting parts from the interviews were added and discussed. The first two authors examined the categories from different perspectives and searched for explicitly stated or concealed pros and cons in relation to how QR are handled in industry. The results from the analysis are found in section 4.

[1] http://serg.cs.lth.se/research/packages

Table 2. Company characteristics

	Alpha	Beta	Gamma	Delta	Epsilon
# employees	~100	~3000	>5000	325	65
Domain	Control systems	Telecom	Telecom	Telecom	Control Systems
Typical project cycle	18 months	48 months	24-36 months	Differs	9 months
#reqs	>1000	~7000	>20000	~100 features,	Differs
#QR	~10% QR	~10% QR	QR unknown	~10% QR	

3.2 Validity

In this section, threats to validity are discussed. We consider the four perspectives of validity and threats presented in Wohlin et al. [27].

Construct validity: The construct validity is concerned with the relation between theories behind the research and the observations. The variables in our research are measured through interviews, including open-ended aspects where the participants are asked to express their own opinions. Mono-operation bias [27] was avoided by collecting data from a wide range of sources on the topic of the study. To avoid evaluation apprehension [27], complete anonymity from other participants, the companies, and researchers was guaranteed. Another validity threat lies in the question that asked interviewees to rank and include additional factors if the list provided to them was inadequate. Interviewees may have thought that it was easier to rank the provided factors than propose new factors. This means that important interdependency types may be missing.

Conclusion validity: Threats to conclusion validity arise from the ability to draw accurate conclusions. The interviews were conducted at different companies and each interview was done in one work session. Thus, answers were not influenced by internal discussions. To obtain highly reliable measures and to avoid poor question wording and poor layout, several pilot studies were conducted.

Internal validity: This threat is related to issues that may affect the causal relationship between treatment and outcome. Threats to internal validity include instrumentation, maturation and selection threats. In our study, the research instrument was developed with close reference to literature relating to non-functional requirements, and influenced by a previously administrated and validated research instrument [18], which mitigates the instrumentation threat. In addition, maturation threats are handled by reducing the duration of interview sessions by collecting background information before the interview, and by keeping the interview session to 90 minutes.

External validity: This threat is concerned with the ability to generalize the findings beyond the actual study. Qualitative studies rarely attempt to generalize beyond the actual setting since it is more concerned with explaining and understanding the phenomena. However, understanding the phenomena may help in understanding other cases. The fact that most of the identified challenges are acknowledged by more than one company increases the possibility to generalize the results beyond this study. To avoid the interaction of selection and treatment, interviewees were selected according to their roles within the company, and companies were selected from different geographical locations.

4 Results and Analysis

This section presents the results discovered during the analysis of the interviews. The five following sub-sections present and discuss one research question each, corresponding to the research questions in Table 1.

4.1 Important Quality Aspects (RQ1)

In analyzing Research Question 1, this section examines the most important quality aspects, as illustrated in Figure 1. Based on Lauesen's comparison of ISO9126 and McCall quality factors [20], we identified 23 different types of QR. We asked the interviewees to rank the top five most important aspects for their products based on their expertise and their own definition of the quality factor. (Our approach was not to impose preconceived definitions but to try to understand existing industrial practice and practitioners' own interpretations of QR.) Looking at Figure 1, the quality aspect ranked first received five points, the one ranked second got four points and so on, and the one ranked fifth got one point. In total we see that interviewees agreed that usability (which got a total of 26 points) and performance (23 points) requirements are the two most important types of QR followed by compliance (13 points), flexibility (13 points), and stability (11 points).

One reason for the prioritization of usability, as explained by several interviewees, is that *"if the product is not usable we will not sell any products"*. One interviewee expanded the view by stating that it does not matter if you have the latest and coolest functionality, if the system is not easy to use, the customer will look at the competitors for an easy to use system. The reason why compliance was ranked as the third most important quality aspect is interesting. Several interviewees explained that compliance is important because *"we must be compliant with the requirement document"*. This interpretation of compliance differs from the one formulated by ISO9126 which states to adhere to standards, regulations and laws. This leads to a possible mismatch between the established academic interpretation of compliance and the industrial interpretation of it. Apart from the agreement that usability and performance were the two most important types of QR, PM and PL had different priorities. PMs ranked performance (14 points) as the most important quality aspect, followed by usability (12 points) and security/integrity (7points).

PLs ranked usability first (14 points), followed by performance (9 points), and compliance and flexibility (8 points). PMs uniquely identified security/integrity, testability, suitability, and installability as the most important quality aspects. On the other hand, PLs uniquely identified the following quality aspects: recoverability, reusability, correctness, and accuracy. The differences in priority between PM and PL may not be a surprise as some mismatch can be expected. The two have different roles and perspectives, but it might nevertheless be an important insight. For example, not a single PL ranked security/integrity among the top five even if security/integrity was considered the third most important by the PMs. Worth observing is that the aspects of fault tolerance, conformance, replaceability, and analyzability was not in any of the PMs or PLs top five.

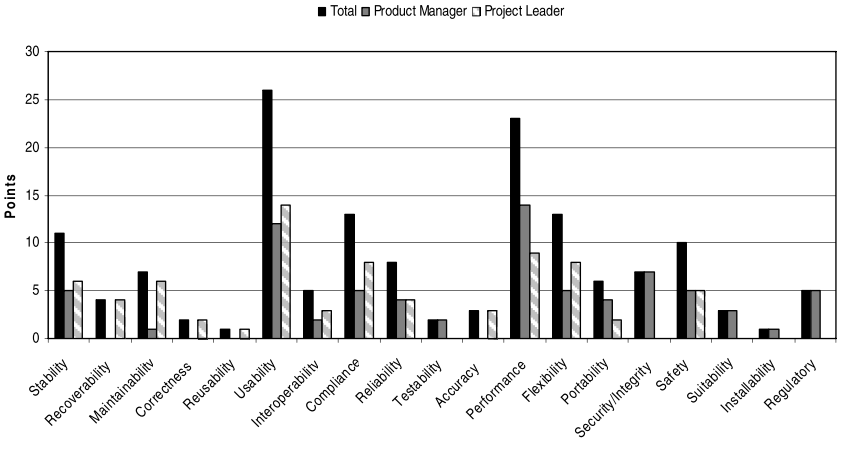


Fig. 1. Importance of quality aspects

4.2 Interdependencies (RQ2)

Six different interdependency types are characterized [4]: **(1)** R_1 *AND* R_2: R_1 requires R_2 to function, and R_2 requires R_1 to function, **(2)** R_1 *REQUIRES* R_2: R_1 requires R_2 to function, but not vice versa, **(3)** R_1 *TEMPORAL* R_2: Either R_1 has to be implemented before R_2 or vice versa, **(4)** R_1 *CVALUE* R_2: R_1 affects the value of R_2 for a customer, **(5)** R_1 *ICOST* R_2: R_1 affects the cost of implementing R_2, and **(6)** R_1 *OR* R_2: Only one of $\{R_1, R_2\}$ needs to be implemented.

Although the interviewees had the option of adding new types of interdependencies, no new types were discovered during the interviews. All of the six presented interdependency types were used by the interviewees to characterize perceived interdependencies, both among different QR, and interdependencies among QR and functional requirements (FR), as illustrated in Table 3.

In general, the most common interdependency types identified among QR were: *OR*, *REQUIRES*, and *ICOST*, while the least frequent one identified was *TEMPORAL*. When the results from PM and PL were examined separately, the findings show a difference of opinion. PMs viewed *OR* and *ICOST* as the most common types, while PLs viewed *REQUIRES* as the most common one.

When examining the most frequent identified interdependency types among QR and FR, four of the six types were considered equally common, while the remaining two (*AND*, *OR*) types were considered least important. However, when examining the results from the PM and PL separately, the findings show an interesting difference. While PM considered *TEMPORAL* as the most common interdependency, PL viewed *TEMPORAL* as least frequent. On the other hand, PL identified *OR* as one of four (the other three are: *REQUIRES*, *ICOST*, and *CVALUE*) equally common interdependency types, but *OR* was viewed as least frequent by PM.

Table 3. Existing interdependency types divided by role

	Role	QR to QR	QR to FR
Alpha	*PM*	REQUIRES, **CVALUE**, ICOST	AND, REQUIRES, TEMPORAL, CVALUE, **ICOST**
	PL	NONE	NONE
Beta	*PM*	OR, AND, REQUIRES, TEMPORAL, CVALUE, **ICOST**	OR, AND, REQUIRES, TEMPORAL, CVALUE, **ICOST**
	PL	OR, **REQUIRES**, TEMPORAL, CVALUE, ICOST	OR, **REQUIRES**, TEMPORAL, CVALUE, ICOST
Gamma	*PM*	OR, AND, **REQUIRES**, CVALUE, ICOST	AND, **REQUIRES**, TEMPORAL, CVALUE, ICOST
	PL	OR, AND, REQUIRES, **CVALUE**, ICOST	AND, REQUIRES, **CVALUE**, ICOST
Delta	*PM*	OR, **ICOST**	OR, REQUIRES, TEMPORAL, CVALUE, **ICOST**
	PL	OR, **REQUIRES**, CVALUE, ICOST	OR, AND, **REQUIRES**, TEMPORAL, CVALUE, ICOST
Epsilon	*PM*	**OR**	**TEMPORAL**
	PL	**REQUIRES**	**OR**

In the study by Carlshamre et al., three of five case companies viewed value related (*ICOST* or *CVALUE*) interdependency types as the most common [4]. In the remaining two cases, functionality related (i.e., *AND* or *REQUIRES*) types were most common. Our results show a mix of value and functionality types as the most common ones (with the exception of Company Epsilon). The difference between the studies may be explained by the focus, i.e.we focused solely on interdependencies related to QR, while in Carlshamre et al. the focus was on requirements in general [4]. In [6, 7], a softgoal interdependency graph (SIG) is used to show interdependencies among QR. The interdependency types used in the SIG are limited to *AND*, and *OR*, which is not inline with the findings in our study, as we found that six different interdependency types were present in the companies. Furthermore, the two types *AND*, and *OR* were only identified as present by 25% of the interviewees.

RQ2.1: What types of interdependencies are deemed most important by practitioners, and is there any difference between the view of product managers and project leaders in this regard? According to the interviewees in total, the most important interdependency type to identify among QR was *REQUIRES*, however, the PM and PL roles were not in agreement. PM considered *ICOST* as the most important, while *REQUIRES* was prioritized by the PL. Interestingly, in identifying the most important interdependency type among QR and FR, the total result was identical to interdependency types among QR. On closer examination the result between PM and PL vary in relation to interdependencies among QR and FR. PM prioritized *ICOST*, but also uniquely identified *TEMPORAL* and *ICOST*. The PL prioritized *REQUIRES*, but also uniquely identified *OR* and *CVALUE*.

It is not surprising that PM and PL have different views on interdependency priority. According to Carlshamre et al., value related interdependencies are subjective; it may be difficult to state whether the cost exceeds the value for the customer, therefore, these types of decisions should be made by product committees [4]. This is inline with the results in this study; PM considers *ICOST* as the most important type, while PL *REQUIRES*. One PL explained that *REQUIRES* is the most important interdependency type because *"functionality first, then the quality aspect of the functionality is relevant"*. Surprisingly, both among QR, and among QR and FR, *REQUIRES* is considered the most important to identify looking at the summation of all interviewees. This result is not inline with Carlshamre et al., which found that *ICOST* and *CVALUE* were the most important types of interdependencies in market-driven developing companies, while *REQUIRES* was considered the most important in bespoke developing companies [4]. One PM explained that *REQUIRES* is considered the most important interdependency to identify because *"this is the easiest type to miss, and therefore the most important to identify"*.

RQ2.2: To what extent are interdependencies elicited, analyzed and documented in industrial practice? The results show that in three of the five companies (Gamma, Delta, and Epsilon) both PM and PL confirmed that no elicitation, analysis, or documentation of interdependencies involving QR was conducted at all. In Company Alpha, the PM stated that all dependency activities were conducted, while the PL from the same company indicated that none of them were performed. In only one company (Beta) both PM and PL stated that activities to elicit, analyze and document interdependencies was performed. This result is inline with results from Karlsson et al., which found that interdependencies between requirements in market-driven software development are a major problem [18]. The problem includes identification, how the requirements affect each other, and how to deal with them. The results are relevant since interdependencies among QR's are at the hart of managing explicit trade-offs among solution alternatives [10]. In addition, Cleland-Huang et al. states that failing to trace QR expose a company to huge risks when a change is introduced [7]. Furthermore, Kamsties et al. found that new requirements may cause unpredictable interaction with existing requirements, which indicates the importance of finding the interdependencies among requirements [17].

There can be several potential explanations of why interdependencies among QR are not actively looked for. Quality requirements tend to have a global impact on the entire system, therefore, QR are difficult to trace and because of the extensive network of interdependencies and trade-offs that exists among them responsibilities for their realization is often vague [6, 10]. Other explanations were discovered during the interviews. Some interviewees stated that they have little focus on QR, while others stated that QR are assumed and therefore interdependencies are not actively looked for. In addition, one interviewee confirmed that their focus is on functional requirements and not QR. Others stated that dependencies are handled during other parts of the development process, for example, during the design, architecture, and implementation. However, they have more focus on functional requirements because functional requirements are easier to discover than QR.

One possible implication with this is that quality aspects such as usability and performance are not considered at the early stages of product and project planning. This can be an acceptable alternative, given that the companies consider quality aspects important only in the solution domain, and not from a product offering or

business perspective. This is however contradicted by the results obtained during the prioritization of quality aspects (see Section 4.1), where the practitioners stated that several (or which usability was premiered) quality aspects were crucial for being able to sell the product at all.

4.3 Quantification of Quality Requirements (RQ3)

In analyzing research question 3, this section examines how often QR are specified in a measurable manner, as illustrated in Table 4.

Table 4. Quantification of quality requirements

Role	**Alpha**	**Beta**	**Gamma**	**Delta**	**Epsilon**
PM	Always	Never	Always	Sometimes	Always
PL	Sometimes	Sometimes	Always	Sometimes	Sometimes

Interestingly, four of the PLs claimed that QR were quantified *sometimes*, while in three of these cases the PMs view differed, stating *always* or *never*. In two out of five (Gamma and Beta) companies agreement between PM and PL could be observed. The disagreement may be an indication of communication problems between the PM and PL. Communication problems were also identified as a challenge in market-driven RE by several studies [12, 13, 18]. In a study by Olsson et al., about half of the QR were found to be quantified which seems to confirm the findings [22]. However, one interesting observation that can not be directly confirmed is the level of disagreement between PM and PL. It should be noted that each PM and PL pair worked for the same company, and moreover with the same project.

4.4 Dismissal of Quality Requirements (RQ4)

We asked the interviewees how often QR that were actually specified and selected for inclusion in a project were subsequently dismissed from project during development (see Table 5). The total average mean value of dismissed QR is 22.5%, meaning almost every fourth QR that has been included in a project is dismissed at some stage. When comparing PM and PL, the least (in the best situation) amount of dismissed QR is slightly higher for PM (5%) than for PL (3%). In worst case (*Most* in Table 5); the mean value of dismissed QR is 55% according to the PMs, while PLs believe that 45% are dismissed.

According to the interviewees, there are two trends of which types of QR that are more representative of the ones being dismissed. Firstly, QR that are not visible for the end customer, such as maintainability and testability are more often dismissed than other QR. Secondly, performance requirements are more often dismissed due to the difficulties of estimating them. One inherent contradiction can be seen in these two trends. For example, if the performance of a system is inadequate, the inadequacy of this quality aspect can be noticed by the customer through a slow system/product. No further elaborations were given on this contradiction.

Table 5. Dismiss rate of quality requirements

	Role	Dismissal rate			Consequence Analysis	Reason for dismiss rate
		Least	*Avg.*	*Most*		
Alpha	*PM*	10%	15%	20%	If customer is affected	Poor cost estimations
	PL	0%	50%	90%	No	Testing QR very late
Beta	*PM*	10%	20%	90%	If customer is affected	Lack of resources
	PL	1%	5%	20%	Yes	Lack of resources and poor cost estimations
Gamma	*PM*	NA	NA	NA	Check with stakeholders	Poor cost estimations and lack of resources
	PL	NA	NA	NA	No	Lack of resources and lower priority than FR
Delta	*PM*	0%	5%	10%	If customer is affected	Issues we cannot affect, e.g. network capacity
	PL	0%	10%	20%	No	Issues we cannot affect, e.g. network capacity
Epsilon	*PM*	0%	50%	100 %	If customer is affected	Poor cost estimations and lower priority than FR
	PL	10%	25%	50%	No	Lower priority than FR

NA: *Not available*

The results reveal three main reasons for the dismissal of QR: (1) poor cost estimations, (2) lack of resources, and (3) that QR have lower priority than functional requirements (FR). Poor cost estimations is related to the difficulties to estimate the cost of QR that have a global impact on the system. The difficulties of estimating the cost of QR are related to lack of knowledge and understanding of how to manage QR in practice. Several interviewees frequently described that QR have lower priority, and that they do not spend much time on managing QR. Some of the interviewees explained that QR are seen as base requirements and therefore not considered. However, this focus has implications on the system, as explained by one PM, *"in most situations, QR are down prioritized by FR due to lack of knowledge of how important a system's quality is. By lowering the quality level, the value of the system decreases"*.

RQ4.1: If QR are dismissed, is any consequence analysis performed? According to the PMs, a consequence analysis is only conducted if the customers are affected. The consequence analysis may include new prioritization of all requirements and new cost estimations, as explained by one interviewee that *"if we have promised a certain quality, then we have to increase the cost for this project and accept a lower return of investment"*. Another consequence, as explained by a PM, is to *"first ask the customer if this is OK. If not, we talk to the developers to find out the reason why this cannot be done. Finally, we decide if we have to add or remove other requirements"*. Surprisingly, none of the PLs shared the view of the PMs. All PL claimed that nothing happens when QR were dismissed from the projects. One explanation, which was qualified by one PL, is that *"we do not have time to re-analyze the consequence of*

QR, other things are more important". Another explanation according to another PL is that *"we can deliver on time if QR are dismissed"*.

A central issue here seems to be the difficulty to properly quantify as well as estimate the cost of implementing a QR, but more importantly the value of a QR. This might indicate a lack of estimation models/techniques for QR. The complexity is of course that a QR often implies a quality aspect of a system/product. Such a quality aspect is often not realized as a feature, but rather implies that all development be in line and adhering to the quality aspect. For example, performance is not dictated by one thing, but often by how the system is realized overall, including architectural considerations impacting the whole.

4.5 Quality Requirement Challenges (RQ5 and RQ6)

In analyzing research questions 5 and 6, this section examines what QR challenges and what QR aspects the practitioners identified. Figure 2 shows the two perspectives.

Three companies (Beta, Gamma, and Epsilon) stated that they are very good in terms of testing QR (QR that are well specified and quantified). This was confirmed by one interviewee: QR that are quantified are easy to test. Another interviewee explained that their company has a well established test organization and good methods for testing QR, both in lab and field environments. However, one of the identified challenges is difficulties in achieving testable QR, i.e. making QR well specified and quantified. This is not a surprising result and is confirmed in previous studies [17, 18, 21]. Apart from the agreement of testing QR, each company identified issues in relation to QR that are adequately handled today. One surprising finding was that one company (Delta) stated that they are good at rejection of QR. The product manager explained that *"we are very good in negotiation of QR, which is to make sure that QR are not part of the contract."*. The result reveals two major challenges that are faced by the companies, (1) how to get QR into the projects, and (2) when is the quality level good enough? All companies faced the same problem of getting QR into the project. The challenge is that QR have to contend with FR, where FR often emerge as victors. Problems with considering QR were also found by Karlsson et al. [18].

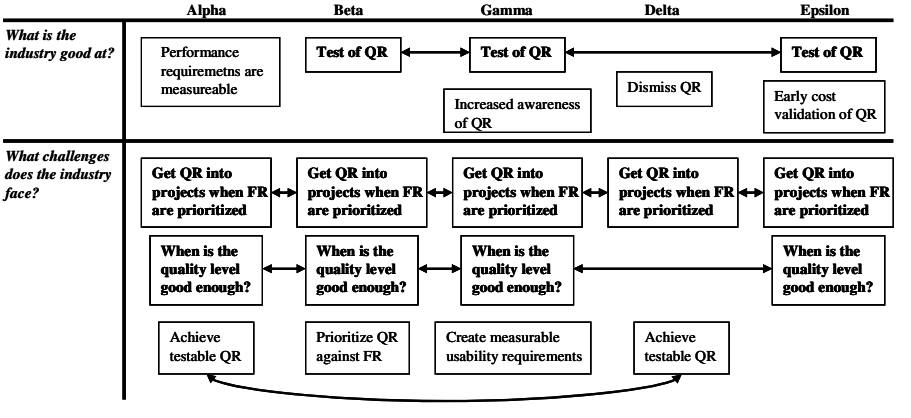


Fig. 2. Challenges and non-challenges in the companies

A reason may be that having an extra function is considered more valuable than to improve the quality of the system. However, this focus may backfire as the customers may want a certain quality level of the systems that are bought. One interviewee confirmed that *"we have been very technology focused, we did not care about QR, but now it has backfired and we have to put a lot of focus on the QR."* In addition, QR are considered as obvious, or even as base requirements and therefore not quantified or specified. The second main challenge is to decide when a certain quality level is good enough, when are you finished with a QR? The interviewees expressed their concern of how to decide when the quality is good enough. Should the performance be two seconds, 1.5 seconds, or even one second, who can decide that? One interviewee said, who can decide if 1 or 5 Mbits are most appropriate?

Two companies (Alpha and Delta) identified achieving testable QR, and one company (Gamma) viewed creating measurable usability requirements as challenges. One reason for identifying these challenges may be related to the quantification of QR (Section 4.3), which shows that 60% of the interviewees stated that QR are *never*, or *sometimes* specified in a measurable manner. Another identified challenge (Company Beta) is prioritization of QR. Prioritization of QR involves other challenges than prioritization of FR, which were further explained by one interviewee by the statement that performance and usability requirements are different in nature and very difficult to compare. How do we prioritize performance requirements of two seconds against the subjective appraisal of usability requirements, was asked by one interviewee.

5 Conclusions

In conclusion, this article presents the results of an empirical study that examines Quality Requirements (QR) in practice in five software companies. Data is collected from five product managers and five project leaders at the companies. To the best of our knowledge, there are no other multi-case survey studies that examine QR in practice.

The findings reveal that usability and performance requirements are deemed the two most important types of QR by the interviewed practitioners. In addition, we found that the companies do not actively look for interdependencies among QR, and we did not encounter QR-specific elicitation, documentation, or analysis (RQ2). The findings highlight three important challenges (RQ5): (1) how to get QR into the projects when functional requirements are prioritized, (2) how to know when the quality level is good enough, and (3) how to achieve testable QR. Our results indicate that QR are often not quantified (RQ3), thus difficult to test. However, the interviewees consider that the companies are good in terms of testing the (few) QR that are quantified (RQ6).

There seems to be a bespoke development mindset where the immediate project gets a higher priority than the long-term evolution of the product (which would be interesting for further studies). This is confirmed by the implicit management of QR, and the dismissal off-hand of QR with little or no consequence analysis (RQ4). This contradicts the interviewees' initial view (RQ1) where quality aspects were labeled as critical, but looking at practice, the project-oriented perspective and the urge of offering more functionality in the immediate release dominates.

The interviewees expressed that the limited focus on QR can have long-term consequences as well; increased maintenance costs and degradation in usability with feature growth are but a few examples. However, the main problem is that QR are not taken into consideration during product planning (pre-project) and thus not included as hard requirements in the projects. This implies that no explicit trade-off can be made, making the realization of QR a reactive rather than proactive effort. Product management may thus not be able to plan and rely on quality aspects to achieve competitive advantages, but mainly respond to emerging QR problems.

References

1. Aurum, A., Wohlin, C. (eds.): Engineering and Managing Software Requirements. Springer, New York (2005)
2. Breitman, K.K., Leite, J.C.S.P., Finkelstein, A.: The World's Stage: A Survey on Requirements Engineering Using a Real-Life Case Study. Journal of the Brazilian Computer Scociety 6, 13–38 (1999)
3. Brooks Jr., F.P.: No Silver Bullet: Essences and Accidents of Software Engineering. Computer 4, 10–19 (1987)
4. Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., Nattoch Dag, J.: An Industrial Survey of Requirements Interdependencies in Software Product Release Planning. In: Proc. 5th IEEE Int. Sypm. on Requirements Engineering, Los Alamitos, USA, pp. 84–91 (2000)
5. Chatzoglou, P.D.: Factors Affecting Completion of the Requirements Capture Stage of Projects with Different Characteristics. Information and Software Technology 39, 627–640 (1997)
6. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Dordrecht (2000)
7. Cleland-Huang, J., Settimi, R., BenKhadra, O.: Goal-Centric Traceability for Managing Non-Functional Requirements. In: Proc. 27th Int. Conf. on Software Engineering, Saint Louis, USA, pp. 362–371 (2005)
8. Curtis, B., Krasner, H., Iscoe, N.: A Field Study of the Software Design Process for Large Systems. Communications of the ACM 31, 1268–1287 (1988)
9. Cysneiros, L.M., Leite, J.C.S.P.: Integrating Non-Functional Requirements into Data Model. In: Proc. 4th IEEE Int. Symp. on Requirements Engineering, Limerick Ireland, pp. 162–171 (1999)
10. Cysneiros, L.M., Leite, J.C.S.P.: Nonfunctional Requirements: From Elicitation to Conceptual Models. IEEE Transactions on Software Engineering 30, 328–349 (2004)
11. Finkelstein, A., Dowell, J.: A Comedy of Errors: The London Ambulance Service Case Study. In: Proc. 8th Int. Workshop on Software Specification and Design, Los Alamitos, USA, pp. 2–4 (1996)
12. Fricker, S., Gorschek, T., Glintz, M.: Goal-Oriented Requirements Communication in New Product Development. In: 2nd Int. Workshop on Software Product Management, Barcelona, Spain (2008)
13. Fricker, S., Gorschek, T., Myllyperkiö, P.: Handshaking between Software Projects and Stakeholders Using Implementation Proposals. In: Sawyer, P., Paech, B., Heymans, P. (eds.) REFSQ 2007. LNCS, vol. 4542, pp. 144–159. Springer, Heidelberg (2007)
14. Gorschek, T., Davis, A.: Requirements Engineering: In Search of the Dependent Variables. Information and Software Technology 50, 67–75 (2008)

15. Gorschek, T., Wohlin, C.: Requirements Abstraction Model. Requirements Engineering Journal 11, 79–101 (2006)
16. Jacobs, S.: Introducing Measurable Quality Requirements: A Case Study. In: Proc. 4th IEEE Int. Symp. on Requirements Engineering, Limerick, Ireland, pp. 172–179 (1999)
17. Kamsties, E., Hörnmann, K., Schlich, M.: Requirements Engineering in Small and Medium Enterprises. In: Proc. Conf. on European Industrial Requirements Engineering, London, UK, pp. 84–90 (1998)
18. Karlsson, L., Dahlstedt, Å.G., Regnell, B., Nattoch Dag, J., Persson, A.: Requirements engineering challenges in market-driven software development – An interview study with practitioners. Information and Software Technology 49, 588–604 (2007)
19. Kusters, R.J., Solingen, R.V., Trienekens, J.J.M.: Identifying Embedded Software Quality: Two Approaches. Quality and Reliability Engineering International 15, 485–492 (1999)
20. Lauesen, S.: Software Requirements – Styles and Techniques. Addison-Wesley, Great Britain (2002)
21. Lubars, M., Potts, C., Richter, C.: A Review of the State of the Practice in Requirements Modelling. In: Proc. 1st IEEE Int. Symp. on Requirements Engineering, San Diego, USA, pp. 2–14 (1993)
22. Olsson, T., Berntsson Svensson, R., Regnell, B.: Non-functional requirements metrics in practice – an empirical document analysis. In: Workshop on Measuring Requirements for Project and Product Success. Palma de Mallorca Spain (2007)
23. Patton, M.Q.: Qualitative Research and Evaluation Methods. Sage Publications, USA (2002)
24. Regnell, B., Höst, M., Berntsson Svensson, R.: A Quality Performance Model for Cost-Benefit Analysis of Non-functional Requirements Applied to the Mobile Handset Domain. In: Sawyer, P., Paech, B., Heymans, P. (eds.) REFSQ 2007. LNCS, vol. 4542, pp. 277–291. Springer, Heidelberg (2007)
25. Robson, C.: Real World Research. Blackwell, Oxford (2002)
26. van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L.: Towards a Reference Framework for Software Product Management. In: Proc. 14th IEEE Int. Requirements Engineering Conference, St. Paul, USA, pp. 312–315 (2006)
27. Wohlin, C., Runeson, P., Höst, M., Ohlson, C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction. Kluwer Academic, Boston (2000)

Does Requirements Clustering Lead to Modular Design?

Zude Li, Quazi A. Rahman, Remo Ferrari, and Nazim H. Madhavji*

Computer Science Department, University of Western Ontario,
London, Ontario, Canada, N6A 5B7
{zli263,qrahman2,rferrari,madhavji}@csd.uwo.ca

Abstract. [**Context and motivation**] The clustering of system requirements groups together related requirements. In a concept paper, we had previously proposed a requirements clustering approach for the purpose of modularizing software. [**Question/problem**] In this short paper, we describe a preliminary study to explore the answer to the posed question: whether or not requirements clustering leads to modular design as measured by design goodness criteria. [**Principal ideas/results**] The study assesses the modularity of software designs developed by independent groups given the same requirements. These are then compared against the expected design resultant from implementing the requirements cluster. [**Contribution**] The study results are encouraging and it warrants further investigation.

Keywords: Requirements Clustering, Design Modularity.

1 Introduction

Requirements clustering is a recognized technique to organize software requirements into a set of clusters with high cohesion and low coupling [4,1,10]. It can also be used in system decomposition [4], modularization [1], software product lines [9], triage [6] and quality improvement [10].

Our previous study [7] has proposed a requirements clustering technique. In this paper, we describe a preliminary empirical validation of this technique. Specifically, we want to explore the answer to the posed question: *whether or not requirements clustering leads to modular design as measured by design goodness criteria*. The study assesses the modularity of software designs developed by independent groups given the same requirements. We used the modularity metric *evolutionary coupling index* (ECI) [11], which is based on measuring coupling and cohesion, as the goodness criteria to evaluate the modularity of the designs developed by these groups. The designs are then compared against the expected design resultant from implementing the requirements cluster derived by using our approach. These comparisons show that our requirements clustering approach can produce good modular design, where the modularity level is close to the modularity level achieved by an expert.

The remainder of the paper is organized as follows: Section 2 briefly reviews our requirements clustering approach [7]. Section 3 describes the study including the preliminary findings. Section 4 discusses future work and concludes the paper.

* This research is, in part, supported by a research grant from Natural Science and Engineering Research Council (NSERC) of Canada.

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 233–239, 2009.

2 Requirements Encapsulation

Requirements encapsulation means organizing requirements into a set of clusters along with external interfaces such that each cluster can be ultimately implemented by a functional module. Its main principle is the Requirement Encapsulation Design Rule (REDR) [8, page3]: *Create one functional module per requirements cluster*.

Our requirements encapsulation approach [7] includes two main steps: *requirements clustering* and *clusters encapsulation*. These are discussed in the next two subsections. The differences between our approach and other requirements clustering techniques are discussed in [7] and so are not repeated here.

2.1 Requirements Clustering

Within our approach, requirements clustering is implemented based on the *Requirement-Dimension-Attribute* (RDA) relations. A *dimension* of requirements depicts an information category of the requirements semantics within the application environment. We decompose requirements into seven organizational or semantic dimensions: *subject, action, object, functionality, quality, time*, and *location*. An *attribute* is the concrete information of a requirement in a dimension. For example, *user* is an attribute of dimension *subject*. Requirements can be decomposed into each dimension as an *attribute*, representing its information granularity level in the dimension. Each attribute is assigned with a *weight* (similar to the *weighting scheme* in [3]) indicating the degree of *similarity* or *associativity* of this attribute with others. In each dimension, attributes are organized in a *hierarchy* with the information granularity.

Example 1: Consider these two requirements, R1: *User may input client data (name, gender, birthdate)*, and R2: *Super-user can input user data (name, password)*. In R1 and R2, *client data* and *user data* can be considered as two attributes in the object dimension, written *Client-Data* and *User-Data*, respectively. They can be taken as two granular specializations of attribute *Data*. Attribute *Client-Data* can be further specialized into three more granular attributes: *Name*, *Gender*, and *Birthdate*, indicating client's name, gender and birthdate data. Attribute *User-Date* can be further specialized into *Name* and *Password* indicating user's name and password. Also the attribute of R1 in the *quality* dimension can be expressed as *Client-Data-Input-GUI*, indicating the quality aspect of this requirement.

The *similarity* and *associativity* degrees are measured between requirements to address the possibility of implementing these requirements in the same module. The *requirements clustering metric* is defined based on these measures.

2.2 Cluster Encapsulation

Requirements cluster encapsulation is to encapsulate each requirements cluster by defining its external interface with a set of *stimulus-response* pairs. The *stimulus* and *response* are defined with the following format:

stimulus = (*source_requirement, trigger*);
response = (*target_requirement, response_behaviour*).

Here *source_requirement* refers to the requirement that launches the *action* (named *trigger*) to another requirement (named *target_requirement*). The possible behaviour of target_requirements responded to the trigger is named *response_behaviour*.

Example 2: Consider requirement R3: *Non-medical entries can be added to client record*. Suppose that R3 and R1 are organized into two clusters. In this situation, R3 has a cluster-level relation with R1, since any non-medical entries to a client record can be added only if the client record has been created. This relation is a temporal sequence constraint which indicates that the action specified in R3 can be possibly accomplished only after the action in R1 is accomplished. We can define the stimulus-response pair on R1 and R3 as follows: (R1, *record-created*)−(R3, *activate-entry-option*), where *record-created* is the trigger identified in R1 that notifies that non-medical entries (in R3) of a client record can be added, as *activate-entry-option*.

After the stimulus-response pairs are defined for each requirements cluster, it is expected to design a module to directly implement a cluster of requirements. The internal requirements attributes and relations in a cluster can be captured as classes (or objects) and relations within a module. The external cluster interface can be addressed by (public) access methods to the module.

3 The Study

The software project used in this study is an open project CAISI (Client Access to Integrated Services and Information), which is developed to help address the problem of homelessness in big cities, by providing integrated information to agencies that help homeless people. The main software requirements have already been posted in the third-party project website (http:/www.caisi.ca/, see also [2]) and so we do not reproduce them in this paper.

We investigated nine CAISI system designs that were developed by nine independent students' groups given the same set of requirements [2] in a university object oriented design course. Each group contained four to six senior undergraduate students. Students did not use any requirements clustering technique during their development process. After the project was completed, we analyzed their system designs using (a) the *evolutionary coupling index* [11] and (b) requirements clusters which are derived using our approach.

In this section, we first describe the clustering of the project requirements [2] using our approach [7] (section 3.1), and then we present some initial study results (section 3.2) to find out how requirements clusters can help to improve design modularity.

3.1 Clustering CAISI Requirements

The clustering task includes the following three steps.

Step 1 (*Dimension hierarchy construction*): For each requirement, we extract its attributes in the seven dimensions (see Section 2.1) and construct the seven dimension hierarchies based on these requirements attributes.

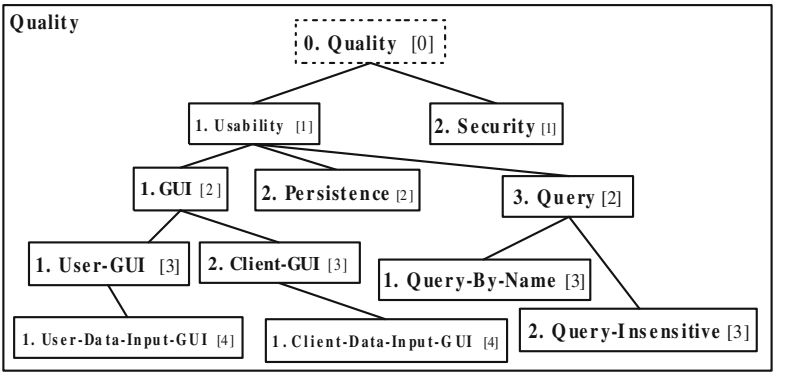


Fig. 1. The *Quality* dimension hierarchy

Example 3: Fig. 1 depicts the Quality dimension hierarchy. *Usability* and *security* are two quality attributes (with weight 1). *Usability* has three refined attributes: *GUI*, *Persistence*, and *Query* (with *weight* 2). The more granular attributes have greater weights.

Step 2 (*RDA building*): The RDA table is built on the seven dimension hierarchies, see Table 1. Each element contains a hierarchical path to an attribute with its similarity or associativity degree in a dimension.

Table 1. The Requirements-Dimensions-Attributes (RDA) table (Partial)

	Functionality	Action	Object	Time	Location	Subject	Quality
R1	1-1-1-1 [4]	1-2-1 [3]	1-1-{1,2,3} [4]	0-1 [1]	1-2 [2]	1 [1]	0-1-1-2-1 [4]
R2	1-1-2 [3]	1-2-2 [3]	1-2-{1,2} [3]	0-1 [1]	1-1 [2]	1-1 [2]	0-1-1-1-1 [4]
R3	1-1 [2]	1-2 [2]	1 [1]	0-2 [1]	1 [1]	1 [1]	0-1 [1]

Example 4: In Table 1, the quality attribute value of R1 is "0-1-1-2-1 [4]", which means the quality attribute of R1 is *Client-Data-Input-GUI* (in Fig. 1, see Example 1) with path *Quality*−*Usability*−*GUI*−*Client-GUI* and the weight (i.e., associativity degree) 4.

Step 3 (*Requirements clustering*): A set of requirements clusters is derived by the requirements clustering algorithm [7] based on the RDA table. For instance, the requirements {R1, R2, R3} (from Example 1 and 2 above) are grouped into a requirements cluster, because they share granular attributions in the seven dimensions.

3.2 Design Analysis

In this study, students' designs of the CAISI system were analyzed. We used the metric *evolutionary coupling index* (ECI) [11] as the goodness criteria for evaluating design modularity. This metric is defined as the ratio of the number of internal dependencies in modules to the number of external dependencies between modules:

$$ECI = \frac{\#Internal\ Couplings}{\#External\ Couplings}\ .$$

It can be used to measure the modularity of the module set with relation to the connections between modules. The greater the ECI, the better the modularity is [11].

Each student group designed the class diagram for the system based on the given set of requirements. They did not do any module level design. To investigate how requirements clusters can help to achieve better design modularity, we compared the design modularity achievable by our clustered requirements to the design modularity achieved by an expert design based on the same set of classes. First the student classes were organized into modules by an expert without considering the requirement clusters. We call this set of modules as Students' modules from Expert (SME). We measured the average modularity level of the SME set using the ECI metric.

To derive modules based on requirements clusters, we identified the traceability links between requirements and classes to find out which requirement is implemented by which classes. For each of the requirements clusters, we organized the linked classes into modules for nine CAISI systems designed by the students. We call this set of modules the Students' modules from requirements clusters (SMRC). Then we measured the average modularity level of the SMRC set using the ECI metric. We used the following metric ECIRatio to compare the modularity levels of the SMRC and SME sets.

$$ECIRatio = \frac{ECI(SMRC)}{|SMRC|} / \frac{ECI(SME)}{|SME|} .$$

The *ECIRatio* measures how close the modularity level of the SMRC set is to the SME set, ranged between 0 and 1. A high *ECIRatio* indicates that the modularity level of the SMRC set (the modules derived from the requirements clusters) is close to the modularity achievable by an expert using the same set of classes.

The results from these measurements are illustrated in Table 2.

Table 2. The design comparison results

GrpNo	#Class	avgECI(SMRC)	avgECI(SME)	ECIRatio
1	12	0.87	0.87	1.0
2	35	0.5	0.61	0.85
3	12	0.31	0.52	0.60
4	21	1.0	1.0	1.0
5	25	1.1	1.37	0.83
6	12	0.34	0.40	0.85
7	27	1.2	1.25	0.95
8	6	1.35	1.5	0.90
9	18	0.7	0.8	0.88

In Table 2, column "#Class" addresses the number of classes in the system, column "avgECI(SMRC)" (or "avgECI(SME)") depicts the average *ECI* level of the SMRC (or SME) set, and column "ECIRatio" represents the *ECIRatio* value. We find that, the average ECIRatio for these groups is 0.87. That means for those groups' designs, our clustering technique could achieve 87% of the design modularity achieved by an expert. In qualitative terms, this indicates that our requirements clustering approach can

produce good modular design where the modularity level is close to that achieved by an expert, as measured by the goodness criteria (*ECI* metric) used in our study.

We also provided the requirements clusters to the expert to do the modular design based on the clusters. We measured the modularity level (*ECI* metric) of the expert modular design that has been derived directly from the requirements cluster. The *ECI* metric value for the design is 1.7 (i.e., the average number of internal couplings is 1.7 times as many as the number of external couplings), which is 100% higher than the average modularity level (the metric value is 0.82) achieved from the students' design.

The above findings are concluded from students' course projects. Although some researchers, e.g., Host et al. [5], have found that student work can be used to conduct empirical studies under certain conditions, we recommend that such studies can be taken a step further when opportunities arise to apply to industry data.

4 Conclusion and Future Work

In this short paper, we have presented a requirements clustering approach and initial results of a study determining the effectiveness of this approach on modular software design. The initial results are encouraging as shown in Table 2. We conclude from this that requirements clustering can lead to modular design. In the future, however, we plan to conduct a controlled study whereby one group would design software using the requirements clusters produced by our clustering approach and another group would design software without using the requirements clusters.

Acknowledgement

We are grateful to Prof. Jamie Andrews of the University of Western Ontario for providing access to the student projects.

References

1. Al-Otaiby, T.N., AlSherif, M., Bond, W.P.: Toward software requirements modularization using hierarchical clustering techniques. In: ACMSE 2005, Kennesaw, GA, USA, pp. 223–229 (2005)
2. Andrews, J.H.: Standalone caisi system (scs): System requirements, version 2.2 (2007), http://www.csd.uwo.ca/courses/cs307b/project/2007/cs307jan2007projectreqs.pdf
3. Andritsos, P., Tzerpos, V.: Information-theoretic software clustering. IEEE Transactions on Software Engineering 31(2), 150–165 (2005)
4. Hisa, P., Yaung, A.T.: Another approach to system decomposition. In: COMPSAC 1988, Chicago, IL, USA, pp. 75–82 (1988)
5. Host, M., Regnell, B., Wohlin, C.: Using students as subjects–a comparative study of students and professionals in lead-time impact assessment. Empirical Software Engineering 5, 201–214 (2000)
6. Laurent, P., Cleland-Huang, J., Duan, C.: Towards automated requirements triage. In: RE 2007, New Delhi, India, October 2007, pp. 131–140 (2007)

7. Li, Z., Rahman, Q.A., Madhavji, N.H.: An approach to requirements encapsulation with clustering. In: WER 2007, Toronto, ON, CA, pp. 97–103 (2007)
8. Lutowski, R.: Software Requirements: Encapsulation, Quality, and Reuse. Auerbach Publisher (2005)
9. Niu, N., Easterbrook, S.: On-demand cluster analysis for product line functional requirements. In: SPLC 2008, Limerick, Ireland, September 2008, pp. 87–96 (2008)
10. Zhang, W., Mei, H., Zhao, H.: Feature-driven requirements dependency analysis and high-level software design. Requirements Engineering 11, 205–220 (2006)
11. Zimmermann, T., Diehl, S., Zeller, A.: How history justifies system architecture (or not). In: IWPSE 2003, Helsinki, Finland, September 2003, pp. 73–83 (2003)

Lessons Learned from Open Source Projects for Facilitating Online Requirements Processes

Paula Laurent and Jane Cleland-Huang

[1] Systems and Requirements Engineering Center,
School of Computing,
DePaul University
{plaurent,jhuang}@cs.depaul.edu

Abstract. [**Context and motivation**] The use of websites for gathering and prioritizing requirements in large-scale distributed projects is becoming increasingly prevalent in the software industry. These websites include both forums and wiki-style collaborative tools, and are designed to allow large numbers of stakeholders to participate in the requirements gathering process. [**Question/problem**] This paper explores and evaluates the forum-based requirements gathering and prioritization processes adopted by vendor-based open source software projects. The findings of this work have implications far beyond the domain of open source projects as they highlight requirements processes that could be applicable to any distributed, web-based requirements process. [**Principal ideas/result**] The effectiveness of various requirements gathering and prioritization practices adopted by vendor-based projects are evaluated, through observing how feature requests are managed in the forums, and also through a survey of vendor-based forum users and project managers. [**Contribution**] Our results highlight practices that could lead to more effective requirements processes in web-based requirements gathering and prioritization tools.

Keywords: large scale requirements, feature requests, requirements elicitation.

1 Introduction

The task of gathering and analyzing requirements is an intrinsically people-centric process in which analysts elicit the needs, wants, and desires from a carefully selected group of key stakeholders. However, as software projects continue to grow in scope, the number of stakeholders involved in the elicitation process tends to rise, making this process difficult to coordinate effectively. Furthermore as projects increasingly extend across geographical and organizational boundaries, it becomes more difficult to organize regular face-to-face meetings meaning that we must rely increasingly on collaborative tools [1].

Most traditional requirements management tools fail to provide adequate support for large distributed projects, even though many of them claim to support a collaborative process. For example, DOORS™ [2] requirements management tool, which is one of

M. Glinz and P. Heymans (Eds.): REFSQ 2009, LNCS 5512, pp. 240–255, 2009.

the industry leaders, is advertised as providing a collaborative environment, but in fact like other similar tools it simply provides a multi-user front-end for entering, updating, and viewing requirements. Although multiple users can work together to construct a Software Requirements Specification, the tool provides no real support for managing large numbers of feature requests, or helping to organize stakeholders into forums where they can work collaboratively to explore their needs and generate requirements. For example, organizations such as NASA routinely manage projects which include thousands of stakeholders using rather basic tools such as CRADLE™. Unfortunately this creates numerous coordination challenges, evidenced by a comment made by a NASA engineer that the paperwork generated in the SpaceStation project could have been used to build a stairway into space, thereby eliminating the need for a rocket launch altogether [3]! One approach to solving this problem involves moving the process online into a forum or wiki.

Open Source Software (OSS) development represents a collaborative community-based effort to develop software in which the users participate in deciding what features to build, and a subgroup of developers participate in designing the solution, writing code, and deploying and maintaining the system [4].

The increasing popularity of OSS, has led to an abundance of open forums in which stakeholders report bugs, discuss issues and request new features. There are currently two common OSS models. The first is the user-based model in which software is developed collaboratively by the users, and in which integration of new features is governed by an executive body. The second model represents a vendor-led approach in which a specific vendor controls the development and integration of new features. Although source code is released to the users to develop additional features, the primary responsibility for development is carried by the vendor.

This paper explores the product enhancement requirements elicitation and prioritization processes of vendor-based open source projects which are quite similar in nature to the kinds of requirements forums that might be used by more traditional projects to help facilitate the requirements process in similar circumstances. Lessons learned from studying both the successes and failures of these elicitation and prioritization models can be applied to future tools designed to support large-scale requirements gathering in non-OSS projects.

2 The Requirements Process

In this section we lay the groundwork for the remainder of the paper by exploring some of the accepted requirements practices of a traditional software development project. The requirements phase is generally comprised of elicitation, analysis, specification, validation and management activities, which can be executed either sequentially or iteratively depending upon the software development lifecycle.

The first activity of elicitation represents a discovery process [5-7] in which analysts gather information about the problem that the proposed new application or feature enhancement will address, identify a core group of stakeholders, discover their needs, emerge and negotiate conflicts and establish clear project scope and boundaries[6,7]. During requirements elicitation knowledge is gathered about the stakeholders' needs by

helping the stakeholders to understand and articulate their problems and describe their own vision of the to-be-developed system [6,7]. Information is iteratively collected, clarified and reformulated [7]. During this phase, the analysts focus on what the system needs to do within the context of its given problem domain while the stakeholders discover how their individual needs fit into the overall project and explore the feasibility of the project as a whole [7].

There are many techniques for eliciting requirements including collaborative sessions such as structured brainstorming sessions and workshops; ethnography, in which members of the development team observe how users interact with an existing application, questionnaires, personal interviews, analysis of existing documentation such as user manuals or problem reports; and finally prototyping in which early quick and dirty GUIs are shown to the users to elicit feedback [6-8]. Conditions such as the type, scope and size of a project and stakeholder availability help to determine the appropriate technique to use [8].

The success of websites and wikis that have been used to gather comments from users, demonstrates that, people willingly take the time to contribute feedback and ideas when given the opportunity. Wikis allow anyone to post their comments and to respond to previous posts. Usually the forums' postings or discussions are displayed in a threaded format which allows everyone to see the discussion unfold. Business analysts or project managers can use wikis or forums to elicit requirements by asking a question, posting their comments in a forum, or by reviewing and participating in the user discussions [1,6]. All of these techniques require the project team, developers and stakeholders to closely communicate and actively collaborate.

In most projects, resource limitations mean that requirements must be carefully prioritized [8,10] using one of several common techniques. Stakeholders may simply place requirements into categories such as mandatory, desirable, or inessential, or else quantitatively rank them [11,12]. More sophisticated methods combine the preferences or decisions made by multiple stakeholders. For example Wiegers proposed an approach in which stakeholders assign each requirement a score from 1 to 9 based on the importance, cost, and technical risk of the requirement [5] and the priority value for the requirement is then calculated as the importance/(cost + risk). A second class of prioritization technique is based on the relative value of requirements and produces a strict prioritization. For example, a Binary Search Tree (BST) can be used to prioritize relatively large sets of requirements. It is constructed by inserting less important requirements to the left and more important ones to the right of the BST. A prioritized list of requirements can then be generated through a depth-first traversal of the completed tree. All of these prioritization methods involve significant levels of participation by the users to rank their needs, wants, and desires, and by the developers to evaluate the cost and risks of the associated development effort.

3 Vendor-Based Open Source Feature Request Forums

All Vendor-based processes that we observed included an open process for gathering and prioritizing requirements. Due to the nature of open source products, this process

was at least partially conducted over the web using an open forum. For the purposes of this study we were interested in discovering the different techniques used by Vendor-based OSS projects to elicit, negotiate, and prioritize software enhancement requirements and to evaluate their effectiveness. Some of these activities involve the general users while other activities are conducted solely by the vendors.

3.1 Requirements Processes in OSS Forums

Most OSS forums follow a similar process for adding and managing feature requests. This process is depicted in Fig. 1. After connecting to the open source project site, a user enters a new feature request into the appropriate forum or performs a topic search to determine if a relevant thread already exists. The user can either browse through various forum threads looking for discussions and comments that are related to their topic of interest, or perform a more structured search of one or more forums by use of keywords or other attributes such as authors' names.

In general, OSS administrators use a variety of techniques to gather new stakeholder feature requests ranging from very passive methods to more proactive ones. Some administrators post details regarding planned releases and ideas for future development in the projects "Announcement" forum, while others not only post questions and solicit stakeholder feedback in the regular forums, but also maintain a dedicated "Feature

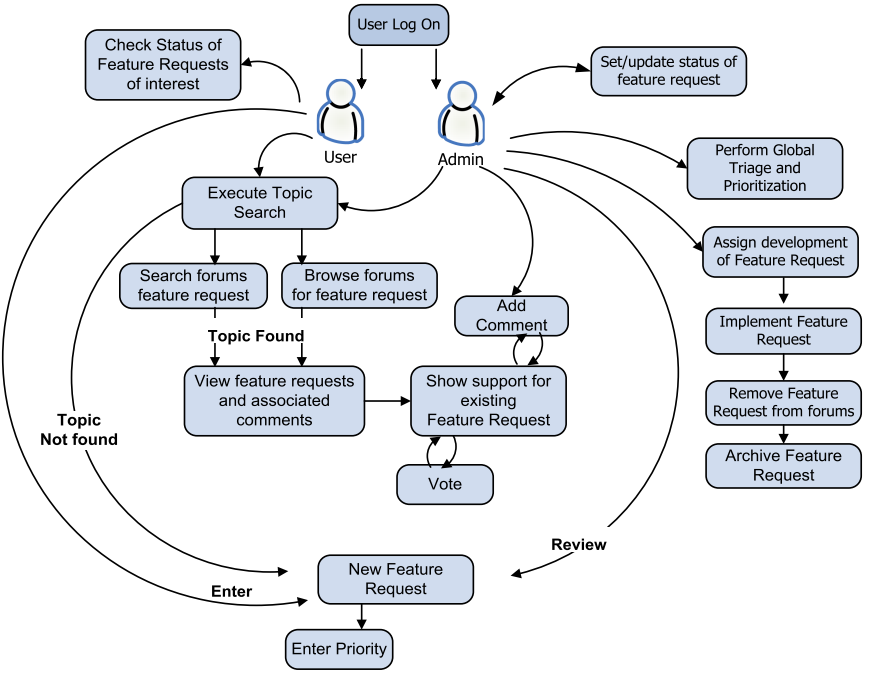


Fig. 1. Vendor Based Open Source Process for Entering and Managing Feature Requests

Requests" forum. New requests are also generated from bug reports and other sub-forums of the issue tracking forum.

The typical OSS administrative process for handling and prioritizing a newly submitted feature request is also shown in Fig. 1. The administrators, or a team of reviewers, review each feature request, determine the feasibility of developing it and prioritize it in relationship to the development team's schedule and resource availability. In some forums users can demonstrate their support of feature requests by casting a vote. These votes are taken into account by the administrators as they attempt to provide the enhancements that will return the most user satisfaction. In many forums, the administrator also is responsible for updating and communicating the status of the request to the users. This process appears relatively straightforward; however our study was designed to evaluate its effectiveness.

Table 1. Number of posts and responses for each of the surveyed forums

Project	Number of posts	User responses	Admin responses
Password Manager	> 17, 000	16	1
Source Code Editor	> 32,000	5	0
File Manager	> 4,000	9	0
ERP	> 20,000	17	1
Java app server	> 440,000	27	1
Virtual world/game	> 2,000,000	10	0
Groupware	> 110,000	4	0
* CRM	> 125,000	0	0

** Included in observational part of study but did not participate in surveys in accordance with administrator's negative response.*

Table 2. Features observed in the Open Source Forums

Legend: A Admin or developer assigns priority; S Stakeholder can assign priority; • Yes	Topic Search		New Feature Request				Dedicated Forum			Status Information			
	Browse	Search	Comment	Prioritize	Vote	Monitor	Feature Request	Issue Tracking	Sort	FR Status Field	FR Process doc	Release Schedule	Remove / Archive Old FRs from Forums
No Predefined Structure of Topics													
Password Manager	•	•	•	A		•	•	•	•	•			
Source Code Editor	•	•	•	A		•	•	•	•	•			
File Manager	•	•	•	A		•	•	•	•	•			
ERP	•	•	•	A		•		•	•	•			
Predefined Structure of Topics													
Java app server	•	•	•	A	•	•		•	•	•			
Virtual world/game	•	•	•	S	•	•		•	•	•	•	•	•
CRM	•	•	•	A			•	•	•	•			
Groupware	•	•	•	A	•	•				•	•	•	

4 Research Approach

In an attempt to include a diverse set of open source projects, we asked software engineering students and IT professional for suggestions about the open source projects in which they participated. The criteria for our exploratory survey were vendor based OSS projects with at least 3000 postings. Fifteen candidate open source project were identified. After further evaluation, the analyses and results of eight projects from a variety of software domains, including games, groupware and system management, were selected for inclusion in the study and are described in this paper.

Our survey questions were geared toward gaining an understanding of the actual activities that users performed when contributing requirements; and the administrator's methodology for eliciting and prioritizing requirements. The survey was reviewed by researchers in our team with industry experience. It was then appraised by several software engineering industry professionals who provided feedback regarding the clarity and appropriateness of the questions. These professionals were similar to our target audience of OSS users. We solicited participation by contacting the project administrators via email. A request for user participation and a survey link were posted in the OSS user forums.

5 The Survey

Our initial analysis of forum based requirements processes, specifically those found in open source software projects, helped us to identify some of the successes and challenges of implementing an online requirements process using a forum. Our study included well-known open source projects in the following industries: a Java application server, password manager software; a source code editor written in C++; a file manager for Windows; a web-based enterprise resource planning tool; a client/server tool for next-generation messaging and collaboration; a virtual world environment/game; and a customer relationship management tool. The characteristics of these projects are summarized in Table 1. Specific project names are not used in order to protect anonymity of the forums.

Observations were made in two different ways. First, we visited each of the forums and analyzed the available tools, adopted processes, and general culture of the forum. Some of the results from this survey are reported in Table 2 and discussed in Section 5. Secondly we conducted a survey of both users and administrators of the identified vendor based OSS forums. Additional OSS projects were included in the study, but one was dropped when we discovered it was actually a community based forum, and the others were dropped when fewer than 4 users responded to the survey. As a result of these surveys we received three responses from administrators and 88 from individual users. Given the low response from administrators, the remainder of this paper discusses their responses only in a qualitative manner. The primary questions which were included in the user survey are shown in Table 3. Additional questions that provided some metadata on the users such as the frequency with which they visited the forums were also asked but not shown here due to space constraints.

Table 3. User survey questions

Questions	Responses
What is your primary role in this OSS community?	Current user of the product
	Prospective user of the product
	Current provider of the product
	Prospective provider of the product
How do you request new features and new functionality?	**<OSS>** Forum @ **<OSS** project forum URL**>**
	Online Newsgroups
	Invited Groups, such as Google, Yahoo, etc.
	Via Email
Which of the following methods do you think your OSSP uses to prioritize feature requests?	Formal voting, by available voting mechanism in the forum.
	User discussions in the forum
	Face-to-face meetings conducted with user groups
	Emails from users
	<OSS> staff members decide which features to build without user input
	Don't know
Who do you think decides which new feature requests to implement in a given release?	A single person (i.e. release manager, project manager)
	A team of people (please explain)
	Other (please explain)
How many feature requests have you made in the last 6 months?	0 requests
	1 request
	2-5 requests
	6-10 requests
	More than 11
Which of these scenarios most closely resembles how you interact with the OSS forums to request new features?	I log in to the forum and type in my new feature request.
	I login to the forum and search to see if my feature request has already been posted by somebody. If I find a similar request I do nothing.
	I login to the forum and search to see if my feature request has already been posted by somebody. If I find a similar request I demonstrate support for it by registering a vote or adding a supportive comment.
How satisfied are you that your feature requests for new functionality are addressed by this process?	Very Satisfied
	Somewhat Satisfied
	Somewhat Dissatisfied
	Very dissatisfied

* Note: Most questions also included options for comments or no response.

6 Forum Observations

Our initial study of forum based requirements processes, helped us to identify several strengths and weaknesses of using forums to support online elicitation and

prioritization processes. The primary strength of the forum approach was its inclusive nature, which enabled large numbers of stakeholders from geographically distributed regions operating in different time-zones to engage in the feature gathering process. Including more stakeholders in the process could increase buy-in to a new product, and could potentially help in the discovery of a complete and correct set of requirements.

We also identified several typical requirements elicitation practices that are difficult to perform in a forum. These challenges can be categorized as follows (i) ineffective processes and tools for bringing the right groups of users together to discuss related needs, (ii) problems in capturing users' priorities, (iii) problems in establishing two-way conversations in which administrators communicate process and decisions, and seek clarification or otherwise engage users in the requirements process, (iv) problems in managing the feature requests in the forum, and finally (v) problems in differentiating between the roles of anonymous users. Each of these problems is explored in more detail in the following four sections.

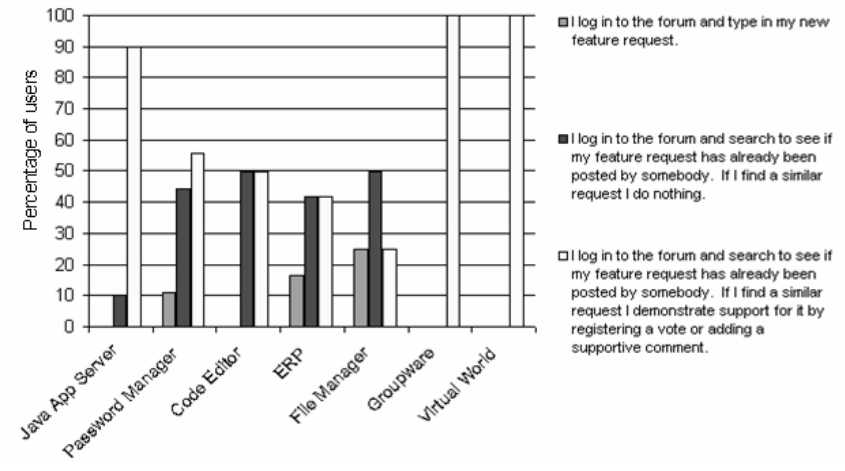


Fig. 2. Methods preferred by users for entering feature requests

6.1 Creating Collaboration

One of the strengths of a more traditional requirements engineering process is that analysts work hard to bring the right stakeholders together to brainstorm ideas and explore their product related needs. Although forums do provide some structure for facilitating this process, our observations show that this task is not accomplished very well.

In a forum, stakeholders engage in discussions by participating in a shared discussion thread. It is the user's responsibility to search for and find appropriate discussion threads. Each of the forums studied provided both browse and search features, primarily designed to help users find relevant discussions. We therefore asked the users whether they searched for relevant topics before entering a new feature request. 12% of users said that they did not perform a search, and just entered

their request into a new thread, while 88% of users claimed to perform a search. These results are depicted in Fig. 2. Nevertheless, a quantitative analysis from a previous study of threads in eight overlapping open source feature-request forums [13], showed that in each forum over 50% of the threads contained only 1 or 2 feature requests. Fig. 3. depicts the distribution of thread sizes in three of these forums; however these results were typical for all forums studied. Further analysis of these threads showed that the majority of them fit into other existing threads, suggesting that users either failed to search for existing threads or searched ineffectively [13].

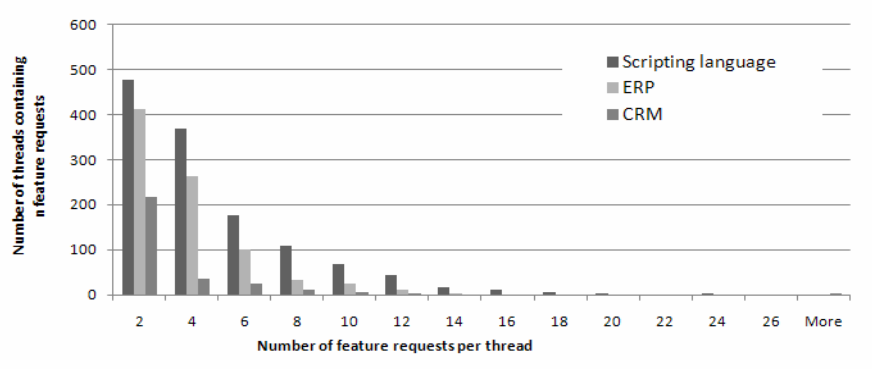


Fig. 3. Distribution of threads sizes in three open source forums

From a requirements perspective, this is a major problem because it means that stakeholders with similar interests do not necessarily engage in a shared discussion of their needs. In fact, two similar or conflicting statements might be placed into separate discussion threads. From a project manager's perspective the fact that individual topics are dispersed across multiple threads, makes the analysis, negotiation and prioritization process very difficult. Browsing support in the forums tended to be very rudimentary and in five of the eight forums studied there was a very flat hierarchy of topics. In the Java App Server, Virtual world, CRM, and groupware forums the administrators provide several coarse-grained categorizations to help organize the forums, which was useful in helping users to find relevant feature requests. In fact one of the surveyed users specifically suggested "*Requests could be categorized: User Interface, Options and Settings, File/Plugin/Feature Support, etc. Having this kind of organization helps people searching for related topics better find their own answers without duplicating requests.*" Similar problems were identified in the use of wikis for requirements engineering. Decker et al noted that the organization of the wiki was improved when administrators created a high-level hierarchy of information [1].

All but one of the forums we studied included a feature for monitoring a discussion thread, so that once a user had found a thread of interest they could be kept informed as additional comments were posted. This was a useful feature as long as the user had already found the correct discussion thread to participate in.

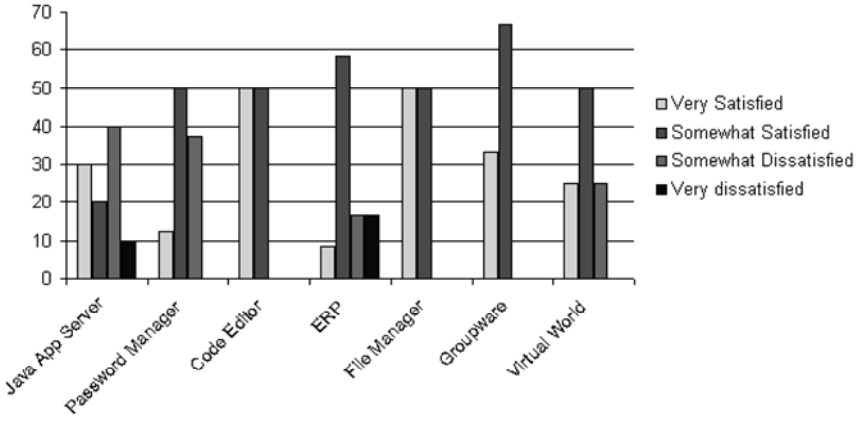


Fig. 4. User Satisfaction with the Requirements Management Process

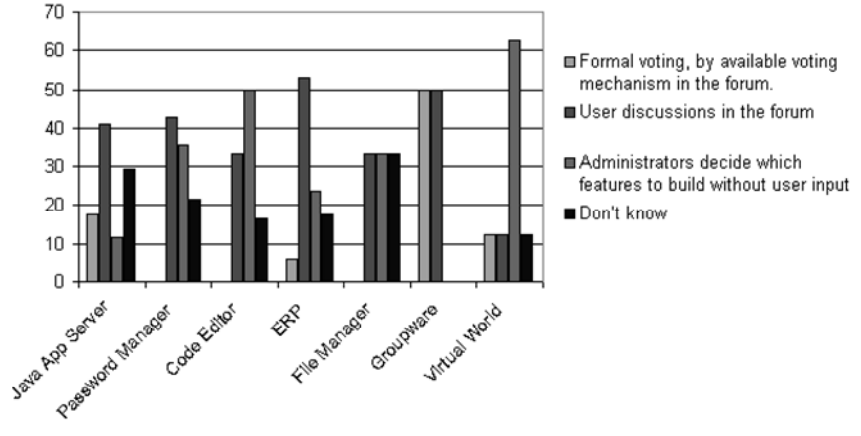


Fig. 5. Methods for prioritizing feature requests

6.2 Prioritization

Three different prioritization methods were observed in forums we visited. The first was a simple voting button that allowed a user to register their support for a feature request. The second method, which was found only in the virtual world game, allowed the contributing stakeholder to assign a priority to a feature request. Finally, all of the forums without more explicit prioritization methods appeared to rely on users expressing their priorities as part of their comments. For example we observed many instances in which users attempted to promote a feature request through adding a comment that included a request such as "*let my comment serve as a vote for this feature*." In forums without dedicated feature request sub-forums, several users resorted to the trick of starting their issue with the words "*Feature Request*" in an apparent attempt to attract attention.

It was clear that users wanted project managers to listen to them and to build features that were important to them. Several forums included posts that made it very clear that the users were either perplexed or annoyed that their favorite requests were seemingly ignored. One of the administrators responding to the question of "*Who decides which new feature requests to implement in a given release*?" responded that "*We have some polls on our website that **might** influence decisions*," which suggests that users' requests are only a small part of the prioritization process.

Another more subtle problem was observed in the virtual world forum which provided both prioritization and voting functions. The initial contributor was allowed to prioritize the requirements, while other users were simply allowed to vote for it. This introduced an ambiguity as to whether users were voting for the feature, or agreeing to its prioritization level. For example, if a contributor had created a new feature request and assigned it a low priority, then subsequent users were unable to change its priority level, although they could cast their votes for it. In general, the forums we observed did not provide sophisticated support for the requirements prioritization process.

In our survey, all three administrators responded that they were only partially satisfied with the requirements prioritization process. User responses to the question "How satisfied are you that your feature requests for new functionality are addressed by this process?" are reported in Fig. 4. Interestingly, the two projects for which users were most dissatisfied were ERP and JavaAppServer also were two of the projects which had no separate feature request module. There was also a significant degree of dissatisfaction in the password manager and virtual world projects. The possible reasons for this are found in users responses to the question "Which of the following methods do you think your OSSP uses to prioritize feature requests?" These responses are reported in Fig. 5. Users were allowed to select as many responses as they wished. The responses indicate that in general, prioritization decisions are made by administrators drawing upon the user discussions in the forums. It was interesting that in the Groupware project, the users' perception was that prioritization decisions were largely based on user input. This project notably had no users that reported being dissatisfied or somewhat dissatisfied with the prioritization method. It should be noted that the level of dissatisfaction by users of the Java App Server, might be correlated to the fact that 30% of the surveyed users did not know how feature requests were prioritized by the administrators.

Users' responses also threw some light on why Virtual World respondents were so dissatisfied with the process. Two respondents who checked the "other" option said that the prioritization was by "*dart game, random selection*", while another user said that "*On rare occasions, there are discussions either in the blog or in the newly-active area in the forums; however, (the administrators) seem to disregard these for the most part although they are the ones who have opened solicited comments*." In fact the virtual world forum provides a webpage describing how feature requests get processed. They include the advice that "*Features are more likely to get implemented if the description of the feature is clear. For a complicated feature, a link to a specification on the wiki is a great way to help flesh out the idea."* Nevertheless, the level of dissatisfaction in the process suggests that users do not believe their feature requests are handled in a satisfactory way despite the appearance of due process.

In CRM's very active discussion forum one of the project managers created a new discussion thread and asked users "*what would you like us to build next?*" In one sense, this demonstrated willingness to engage the user base in the prioritization process, but in another sense it demonstrated failure to extract user priorities out of the active discussions which contained numerous feature requests. This highlights the challenges of conducting the requirements capture process using a web-based environment, where staggering amounts of data need to be processed in order to extract useful information. It seems that despite the active discussions in many of the forums, administrators are still not easily able to understand the users' real needs.

6.3 Engaging and Communicating

Our study identified four primary techniques by which vendors communicated with users in order to engage them more proactively in the requirements elicitation process. First, in several forums we found that administrators and project managers actually participated in the discussion threads. All of the forums we studied included a status field that was used to communicate the status of each feature request. Two of the forums had published processes, although interestingly one of them, the virtual world, was the forum for which users gave the most negative feedback about their prioritization process. The published process included a description of steps a user should take to get their feature requests noticed; however the general consensus by users of this particular forum was that the administrators largely ignored users' requests and built whatever features they felt inclined to build. The same two forums that published process descriptions also posted release schedules in which feature requests had been copied from the primary forums and ranked in order of their likely release.

Our observations of these forums led us to conclude that most forum administrators saw the forums as a means to eliciting information that might be considered in the requirements prioritization process. Notably absent from any of the forums however were the type of questions that analysts usually engage in during the requirements process to clarify and explore the needs of the users. We found few examples of project administrators asking users to explain something in more detail, although there were numerous peer-to-peer examples of this. This problem may be recognized by project administrators. For example, one of the administrators stated that "*We would like more involvement from the community and are experimenting with various tools to elicit more feedback.*" Incidentally, this comment was made by the administrator of the ERP forum, which exhibited the highest level of dissatisfaction in responses to our user survey.

6.4 Managing Feature Requests

Our study also unearthed a number of problems related to managing feature requests. For example we observed that most forums had no way of removing feature requests from the forums once they had been either implemented or designated as non-implementable. There were numerous occasions in the forum discussions that we observed users frustrated because they thought that features they had requested had gone ignored, while in fact those features had been released in recent versions of the product.

Seven out of the eight forums we observed included issue tracking features including feature status fields, and sorting features that users could use to check up on the status of their requested feature requests; however only the virtual world forum had a method in place for removing feature requests from the forum once they were implemented, and also of archiving old feature requests. None of the forums had methods for reporting back to the user if a specific feature request was not considered feasible for implementation.

The three administrators surveyed made some suggestions for improvements. One administrator suggested that "*We should remove feature requests that obviously never will be implemented, even though they are good ideas. Keeping a long list of feature requests that will never be implemented only disappoints users*", while one user requested that "*There should be a website where new features are listed, documented and prioritized so the users can determine how possible it is for them to actually be implemented. This doesn't change the way in which the feature requests are handled but informs the prospective users of them."*

In general, almost all of the forums we surveyed did a very poor job in managing the status of each feature request. For example, feature requests that were never implemented, generally languished in the discussion forum, and every now and again a user would complain that the feature was not implemented. Unfortunately, none of the forums we surveyed had any means of communicating that a given feature request would not be implemented. Furthermore, old discussion threads for features that had already been implemented were rarely removed from the forum. In most cases, when discussion threads were either sorted chronologically or according to activity level, old feature requests tended to drift to the bottom of the list. None of the forums provided traceability between old feature requests and the releases in which they were actually implemented, and so a user searching the forum might easily believe that they had not yet been implemented.

6.5 Role-Based Elicitation

Although one of the intrinsic strengths of online forums is their ability to elicit needs from any stakeholder, this is also a major limitation because all of the forums we studied did not differentiate between different users. One administrator specifically said that an area of improvement would include "*getting feedbacks directly from organizations using our product and then going over them and finding common denominators.*" To implement this type of differential prioritization requires forums to improve their registration process so that the true role and affiliation of users are known.

7 Lessons Learned

The study reported in this paper highlights a number of interesting lessons that can be learned and applied in future forum-based requirements processes. First of all, forums must provide better support for grouping related feature requests together, so that users with similar interests are placed into the same discussion group. The findings from this study suggest that making users solely responsible for the

placement of their own feature requests into threads, will result in a proliferation of redundant discussion threads, and topics dispersed over multiple threads. Our proposed solution to this problem is two-fold. First, data-mining methods can be used to help manage the creation of new discussions in order to prevent the creation of redundant discussion topics [14-16]. Secondly, project administrators can create a predefined hierarchy of topic structures [1].

Project managers also need to increase communication between themselves and the users of the forums. There needs to be higher visibility concerning the status of each feature request, and a clearer definition of the process by which features are prioritized. Project managers also need to more actively engage in the forum discussions in order to truly understand the stakeholders' needs through asking meaningful questions. This of course is only useful if the project manager's intention is to carefully weigh the input of the stakeholders and deliver a product which is designed to meet their needs. Conducting the requirements elicitation and prioritization process in a forum or wiki setting, should not be seen as an opportunity to return to passive requirements gathering methods. Analysts and project managers should actively participate in the process, by visiting and participating in discussion threads, and requesting clarifications when needed.

Prioritization mechanisms must also be improved so that project managers can easily understand the current priorities of their users. This introduces the additional need to track roles and affiliations for each of the users so that prioritization decisions can take this type of information into account. The current comment-based prioritization methods found in most of the forums we studied does not allow the project manager to query the forum for specific priorities. Voting and other prioritization mechanisms need to be significantly improved so that all users are able to provide weighted priorities for each feature request, so that administrators can issue queries that return meaningful insights into the users' needs. Furthermore, as users are unlikely to be available during actual prioritization meetings, they should be provided tools for entering rationales behind their feature requests and their prioritization requests.

Finally, forums also need to be restructured so that it is possible to differentiate between feature requests, comments, and issues. The best approach we observed was to have specific forums dedicated to feature requests, and to create a hierarchy so that comments can be associated with each feature request. Each feature request needs to pass through a distinct lifecycle in which it is created, prioritized, scheduled or deferred, and then implemented or marked as a feature that will not be considered for implementation in the near future. Feature requests that are satisfied must be removed to a separate webpage and must be explicitly traced to the release in which they have been implemented.

8 Conclusions

The primary purpose of conducting this study was to gain some understanding of how forums should be designed to provide increased support for the requirements engineering process. The distributed and asynchronous nature of Vendor-based open-source software projects has naturally led to the use of forums to capture feature

requests. Unfortunately as this study has shown, these forums suffer from a number of problems that inhibit the use of many normally accepted and successful requirements engineering practices. With the increasing trend towards using both forums and wikis as part of more traditional software development efforts it is critical to identify these problems and address them in the second generation of forums designed to capture and manage feature requests.

Given the apparent trend towards online collaborative requirements engineering environments, the challenges outlined in this paper should prove useful in helping us to improve the functionality and utility of future online requirements forums. Our ongoing work in this area focuses on building data-mining tools to augment the process of online, large-scaled requirements processes through automating topic discovery [15] and by using recommender systems to keep users informed of relevant discussions [14]. Addressing these problems will help to improve online requirements processes.

Acknowledgments

We would like to thank Phik Shan Foo for her help in developing some of the graphics for this paper.

References

1. Decker, B., Ras, E., Rech, J., Jaubert, P., Rieth, M.: Wiki-Based Stakeholder Participation in Requirements Engineering. IEEE Software 24(2), 28–35 (2007)
2. Telelogic Products. Increase quality with Requirements Management and traceability (Retrieved December 4, 2007), http://www.telelogic.com/products/doors/index.cfm
3. Hooks, I.F., Farry, K.: Creating Successful Products Through Smart Requirements Management. Amacon, New York (2001)
4. Scacchi, W.: Free/Open Source Software Development: Recent Research Results and Emerging Opportunities. In: 6th Joint Meeting on European Software Engineering conference (ESEC) and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE) 2007, Dubrovnik, Croatia (2007)
5. Wiegers, K.E.: Software Requirements, 2nd edn. Microsoft Press, Redmond (2003)
6. Robertson, S., Robertson, J.: Mastering the Requirements Process. Addison-Wesley, Reading (1999)
7. Sawyer, P.: The Context of Software Requirements. In: Thayer, R.H., Christensen, M.J. (eds.) Software Requirements, The Development Process, 3rd edn., vol. 1. John Wiley & Sons, Chichester (2005)
8. Davis, A., Dieste, O., Hickey, A., Juristo, N., Moreno, A.: Effectiveness of Requirements Elicitation Techniques. In: IEEE International Requirements Engineering Conference, Minneapolis, MN, September 2006, pp. 179–188 (2006)
9. Davis, A.M.: The Art of Requirements Triage. IEEE Computer 36, 42–49 (2003)
10. Karlsson, J., Ryan, K.: A Cost-Value Approach for Prioritizing Requirements. IEEE Software 5, 67–75 (1997)
11. Karlsson, J.: Towards a Strategy for Software Requirements Selection, Licentiate Thesis 513, Department of Computer and Information Science, Linkoping University (1995)

12. Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley, Reading (2000)
13. Cleland-Huang, J., Dumitru, H., Duan, C., Castro-Herrera, C.: Automated support for managing feature requests in open forums. Communications of the ACM (2009)
14. Castro-Herrera, C., Duan, C., Cleland-Huang, J., Mobasher, B.: A Recommender System for Requirements Elicitation in Large-Scale Software Projects. In: ACM Symposium on Applied Computing, Data Mining track, Honolulu, HI (to appear) (March 2009)
15. Duan, C., Cleland-Huang, J., Mobasher, B.: A Consensus Based Approach to Constrained Clustering of Software Requirements. In: ACM Conference on Information and Knowledge Management, NAPA, CA, USA, October 2008, pp. 1073–1082 (2008)
16. Laurent, P., Cleland-Huang, J., Duan, C.: Towards Automated Requirements Triage. In: 15th IEEE International Requirements Engineering Conference (RE 2007), Delhi, India (2007)

Author Index