19th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2013)

In: ICB Research Reports - Forschungsberichte des ICB / 2013

DOI: https://doi.org/10.17185/duepublico/47034

URN: urn:nbn:de:hbz:464-20180914-144539-1

Link: https://duepublico.uni-duisburg-essen.de/servlets/DocumentServlet?id=47034

Source: ICB-Research Report No. 56, July 2013

ICB
Institut für Informatik und Wirtschaftsinformatik

Richard Berntsson Svensson, Daniel M. Berry, Maya Daneva, Joerg Doerr, Sergio España, Andrea Herrmann, Georg Herzwurm, Anne Hoffmann, Raul Mazo Pena, Andreas L. Opdahl, Óscar Pastor, Wolfram Pietsch, Camille Salinesi, Kurt Schneider, Norbert Seyff, Inge van de Weerd, Roel Wieringa, Krzysztof Wnuk (Eds.)

19th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2013)

ICB-RESEARCH REPORT

Proceedings of the REFSQ 2013 Workshops CreaRE, IWSPM, and RePriCo, the REFSQ 2013 Empirical Track (Empirical Live Experiment and Empirical Research Fair), the REFSQ 2013 Doctoral Symposium, and the REFSQ 2013 Poster Session

UNIVERSITÄT DUISBURG ESSEN

Open-Minded

ICB-Research Report No. 56
July 2013

Die Forschungsberichte des Instituts für Informatik und Wirtschaftsinformatik dienen der Darstellung vorläufiger Ergebnisse, die i. d. R. noch für spätere Veröffentlichungen überarbeitet werden. Die Autoren sind deshalb für kritische Hinweise dankbar.

The ICB Research Reports comprise preliminary results which will usually be revised for subsequent publications. Critical comments would be appreciated by the authors.

Contact:

Institut für Informatik und
Wirtschaftsinformatik (ICB)
Universität Duisburg-Essen
Universitätsstr. 9
45141 Essen

Tel.: 0201-183-4041
Fax: 0201-183-4011
Email: icb@uni-duisburg-essen.de

ISSN 1860-2770 (Print)
ISSN 1866-5101 (Online)

Abstract

This ICB Research Report constitutes the proceedings of the following events which were held during the Requirements Engineering: Foundation for Software Quality (REFSQ) conference 2013 in Essen, Germany.

- Creativity in Requirements Engineering (CreaRE)
- International Workshop on Requirements Prioritization and Communication (RePriCo)
- International Workshop on Software Product Management (IWSPM)
- Alive Empirical Study
- Online Questionnaires
- Empirical Research Fair
- Doctoral Symposium
- Poster Session

Table of Contents

Part I: REFSQ 2013 Workshop Proceedings

1 Preface 3

2 Creativity in Requirements Engineering (CreaRE) 7

3 International Workshop on Software Product Management (IWSPM) 51

4 International Workshop on Requirements Prioritization and Communication (RePriCo) 101

Part II: REFSQ 2013 Empirical Track Proceedings

5 Preface 147

6 Alive Empirical Study 153

7 Online Questionnaires 159

8 Empirical Research Fair 183

Part III: REFSQ 2013 Doctoral Symposium Proceedings

9 Preface 193

10 Doctoral Symposium 197

Part IV: REFSQ 2013 Poster Session

11 Preface 241

12 Poster Session 245

Part I

REFSQ 2013 Workshop Proceedings

1 Preface

Editors

Raul Mazo Pena
Université Paris 1 Panthéon Sorbonne, France, raul.mazo-pena@univ-paris1.fr

Camille Salinesi,
Université Paris 1 Panthéon Sorbonne, France, camille.salinesi@univ-paris1.fr

REFSQ 2013 Workshops Proceedings

Preface

Camille Salinesi, Raul Mazo Pena

Université Paris 1 Panthéon Sorbonne
{Camille.Salinesi, Raul.Mazo-Pena}@univ-paris1.fr

Before becoming a Working Conference, REFSQ was itself a workshop. As such, it was organized as a forum of discussion for presenting ground breaking ideas, as well as exchanging on new research problems, or reporting and discussing results of more mature research. The "magic potion" that made REFSQ a successful workshop is now used for "REFSQ workshops", i.e. workshops associated to the RESQ Working Conference: pre proceedings that allow participants to read each other's papers and prepare their questions in advance, post-proceedings that give a possibility to revise papers in light of discussions held during workshops, mixed audience of senior researchers and industrials with young promising researchers, and promotion of interactivity between participants during the workshops as well as during the conference, e.g. by reporting workshop results at the conference, etc. It is now the third year that REFSQ organizes co-located workshops. This experience is very positive, both because it helps extending and strengthening the RE community, but also and this is very important- because it is a unique opportunity to deal with a variety of specific requirements related issues with a very large and diverse audience.

The REFSQ 2013 call for workshops proposals aimed at workshops that had the potential to significantly advance requirements engineering. Many topics were identified, but the call was open to any topic that was important for practice, new to the field, or had controversial viewpoints. 3 workshops were accepted after a thoughtful selection process:

- CreaRE'13, the Third Workshop on Creativity in Requirements Engineering, organised by Maya Daneva, Andrea Herrmann, Anne Hoffmann, and Kurt Schneider
- IWSPM'13, the Seventh International Workshop on Software Product Management, organised by Richard Berntsson Svensson, Inge van de Weerd, and Krzysztof Wnuk, and
- RePriCo´13, the Fourth International Workshop on Requirements Prioritization and Communication, organised by Benedikt Krams, and Sixten Schockert.

We hope that, as we observed since years at REFSQ, the dialogue among participants will lead to interesting follow-up research collaborations, empirical investigations and industrial practices, and of course more workshops and more publications!

We hope you will enjoy these papers, and some day, live the thrill of the "I heard it first at REFSQ!" experience.

Camille Salinesi and Raul Mazo
REFSQ2013 Workshop Chairs

2 Creativity in Requirements Engineering (CreaRE)

Editors

Maya Daneva
University of Twente, Netherlands, m.daneva@utwente.nl

Andrea Herrmann
Software Engineering Trainer and Researcher, Germany, herrmann@herrmann-ehrlich.de

Anne Hoffmann
Siemens AG, Germany, anne.hoffmann@siemens.com

Kurt Schneider
Leibniz Universität Hannover, Germany, kurt.schneider@inf.uni-hannover.de

Workshop Programme

CreaRE 2013 3rd Workshop on Creativity in Requirements Engineering 9
Maya Daneva, Andrea Herrmann, Anne Hoffmann and Kurt Schneider

On the Sizes of Groups Using the Full and Optimized EPMcreate Creativity Enhancement Technique for Web Site Requirements Elicitation 15
Victoria Sakhnini, Luisa Mich and Daniel M. Berry

Requirements from the Void: Experiences with 1:10:100 31
Koen van Turnhoutγ, Stijn Hoppenbrouwersγ, Paul Jacobsγ, Jasper Jeurens, Wina Smeenk and René Bakkerγ

The Even Darker Side of Creativity Why Software Testers Should Check on Requirements 41
Hans Hartmann

CreaRE 2013
3rd Workshop on Creativity in Requirements Engineering

Maya Daneva[1], Andrea Herrmann[2], Anne Hoffmann[3], Kurt Schneider[4]
[1] University of Twente, Netherlands, m.daneva@utwente.nl
[2] Software Engineering Trainer and Researcher, Germany, herrmann@herrmann ehrlich.de
[3] Siemens AG, Germany, anne.hoffmann@siemens.com
[4] Leibniz Universität Hannover, Germany, kurt.schneider@inf.uni hannover.de

1 Technical Program

The CreaRE workshop at REFSQ 2013 took place as a half-day workshop on April 8. The program included a keynote talk, three paper presentations, a discussion on the topic of "Experiences with Creativity Techniques in RE and Research Questions", and an art exhibition. The following speakers were involved:

- *Chris Rupp* (keynote): Gyro Gearloose's Heirs - Going Nuts for Progression?
- *Victoria Sakhnini, Luisa Mich, Daniel M. Berry:* On the Sizes of Groups Using the Full and Optimized EPMcreate Creativity Enhancement Technique for Web Site Requirements Elicitation
- *Koen van Turnhout, Stijn Hoppenbrouwers, Paul Jacobs, Jasper Jeurens, Wina Smeenk, René Bakker:* Requirements from the Void: Experiences with 1:10:100
- *Hans Hartmann:* The even darker side of creativity

2 Introduction

Requirements Engineering (RE) demands a systematic approach for eliciting, operationalizing, and documenting requirements and for solving their conflicts. However it also is a creative activity. It demands the stakeholders to create visions of future software systems and to imagine all their implications. Creativity-enhancing techniques, which have been developed and used in other disciplines and areas of problem-solving, have the potential to be adapted and adopted in today's RE, and thus become the foundation for innovative RE processes, addressing both problem analysis and solution design.

The CreaRE 2013 workshop brought together requirements engineering professionals from industry and researchers who are interested in discussing the role of creativity in RE, the array of creativity techniques that can be applied to RE, and the specific ways to do so. The workshop leveraged the discussion-intensive format and style that was established in the previous two workshop editions (2010 and 2012) to exchange experiences and research results among the participants. In line with

CREARE's mission to raise awareness in the RE community for the importance of creativity and creativity techniques, the CREARE 2013 edition of the workshop reached out and made a first step towards linking the RE community to other communities to which creativity is essential.

We invite interested readers to review the CreaRE 2013 web site for further information: http://www.se.uni-hannover.de/events/creare-2013

3 Program Committee

We are indebted to our program committee members for their continual support:

Dan Berry	University of Waterloo, Canada
David Callele	University of Saskatchewan, Canada
Joerg Doerr	Fraunhofer Institut IESE, Germany
Letícia Duboc	State University of Rio de Janeiro, Brazil
Cathy Ennis	University of Utrecht, The Netherlands
Abdelkader Gouaich	University of Montpellier, France
Thomas Herrmann	Ruhr-University of Bochum, Germany
Heather Niven	Science City York, UK
Klaus Schmid	University of Hildesheim, Germany
Roel Wieringa	University of Twente, The Netherlands
Konstantinos Zachos	City University London, United Kingdom

Each of the submitted papers was reviewed by three program committee members. The acceptance of any contribution was based on these reviews. Before the workshop, the authors of accepted papers revised their papers, taking into consideration their reviewers' comments. After the workshop, they had the opportunity to take into account the feedback that they received during the workshop's discussions.

4 Art Exhibition

A new feature of CreaRE 2013 was the art exhibition. This year, for the first time, CreaRE invited the paper authors and program committee members to present results of their creative activities in an art exhibition. We were able to enjoy the following exhibits:

- Several really professional photos by Chris Rupp (see also on: http://www.chrisrupp.net/)
- A specification and sample of a perfect bagel satisfying all requirements, by Daniel Berry (see also: https://cs.uwaterloo.ca/~dberry/#RiseAndShine)
- Some short lyrics by Andrea Herrmann

Unfortunately, the hotel could not provide us with a piano. Otherwise, Hans Hartmann would have given a concert during the coffee break!

5 Keynote Presentation: “Gyro Gearloose's Heirs - Going Nuts for Progression?” by Chris Rupp

Chris Rupp started her key note talk with shocking news: Practitioners do not use creativity techniques. So, no field report about creativity techniques in practice is possible. It is not techniques that create new ideas and innovative products, but it´s people! Chris Rupp discussed the following questions: Where do ideas come from? How do people form ideas and concepts?
The speaker presented theories and perspectives on innovation and creativity from multiple disciplines. The CreaRE 2013 participants learned about the creative process used in fields as civil architecture design, photography, art artifact design. This process includes four phases: In the **preparation** phase, information is gathered. The **development** phase is a phase of seeming inactivity where ideas are allowed to form. Then, suddenly, the **enlightenment** happens: The idea is here! In the **verification and execution** phase, the idea´s usefulness is evaluated and if it is good, it is applied. New ideas do not only arise at the workplace, but ‘happen’ everywhere and anytime.
Below, we present examples of some perspectives included in the presentation:

- According to Lewin´s psychological equation, behavior is a function of both person and environment. The same person might behave differently in another environment and different persons might behave differently in the same environment.
- Andy Hunt (“Pragmatic Thinking and Learning”) says that we have two different brain modes: The linear and slow one, and the non-linear, fast one. The latter creates the ideas, the former makes them come true. For being creative, formal methods must be avoided. Instead, ideas are intuitively created by unusual perspectives.
- Bruce Barnbaum („Die Kunst der Fotografie“) on the contrary thinks that creativity arises from intelligence. For him, creativity is a conscious process based on thinking and re-thinking. Good ideas are what is left after all useless ideas have been rejected.
- Rob Austin and Lee Devon („Artful Making: What Managers need to know about how artists work”) have published a list of 10 beliefs about creativity. These beliefs, they compare with reality and show that they contain (only) a grain of truth. The first one is that creative ideas emerge mysteriously from the unconscious. Furthermore, it is not generally true that creativity is higher when conventions are rejected, from outsiders, when people are alone, that creativity is a personal trait and that creativity and mental illness are related.
- Gunther Dueck (“Das Neue und seine Feinde: Wie Ideen verhindert werden und wie sie sich trotzdem durchsetzen“) discusses the problems that ideas have when they meet reality.
- DeMarco and Lister („Peopleware“) claim that creativity in an organization is promoted by team work and communication, aside from daily work.

The keynote ended up with a number of open questions to the CREARE 2013 participants on what they think the creative process in their organizations was.

6 Discussion on Experiences with Creativity techniques in RE and Research questions

The topic of the discussion was chosen to leverage the expertise by the expected high number of senior specialists, both practitioners and researchers attending the workshop. CreaRE 2013 benefited from their participation and perspectives shared.-

The discussion included two central questions: what research questions are industry-relevant and what academic researchers can do to help industry find good answers? The first part of the discussion brought an interesting and a surprising result: in the view of our CreaRE 2013 practitioners, the research questions should not come from them but should be defined from the academic researchers. Practitioners perceived themselves as the 'customers' using creativity techniques in RE. They thought of themselves as users of application software who are supposed to use the system but who more often than not are unable to precisely state their user requirements. In light of this view, how could these practitioners possibly know the truly critical research questions? (This is similar to the question: Do customers know the requirements which the software must satisfy?)

This finding has an important implication to RE researchers. To them, the finding means that they should actively keep in touch with industry at all times, follow trends, understand hot issues, major problems and get to know the 'stakeholders' or 'the customers' of the creative techniques much better, than this is happening right now.

The regular discussion of researchers and practitioners will jointly generate ideas.

At CreaRE 2013, this finding helped us reframe the discussion and posing the questions of what types of activities are necessary on researchers' side so that the output is meaningful to practitioners. The audience found the following themes worthwhile taking actions upon:

- Comparison of creativity techniques: Do different creativity techniques create different types of ideas? Do various techniques differ in getting more or less useful ideas?
- How to measure quality and appropriateness of a creativity technique in RE? Where to point creativity at? The absolute number of ideas created definitively is not the main measure for the quality of a creativity session. Important is that there were many good and useful ideas. However, the usefulness of ideas can only be judged after some time. So far, no metrics and no way of measuring ideas´ quality are known, especially with respect to requirements created. Which creativity technique is most appropriate in which situation? This depends on the people involved in the session, the topic, the expert moderating the session and probably more factors. Which of these factors determines which creativity technique is most appropriate? How can creativity techniques be customized for a specific situation?
- How many ideas do we need? How do we know that we can stop the creativity phase?
- How can the really creative, innovative ideas be identified? This question was especially asked with respect to the complex undertaking of system-of-systems development.
- In a creativity session, one works with constraints and real people. Not all factors of success can be controlled. Therefore, in practice people start with

methods that work. Knowledge about "what works" is implicit and there are no studies about this implicit experience knowledge.

- What RE specialists can learn from software architects and creativity? When involved in systems-of-systems projects, software architects tacitly apply some creativity-based approaches to come up with a suitable architecture, the solution design to the problems identified in the early requirements stage. What do software architects do and how do they use their creative energy? And can parts of what they do be applied to RE? There is a difference between brand new ideas and incremental improvements. Do some techniques create more new ideas? How to measure the newness of an idea with respect to the problem and with respect to the state of the art?
- Creativity is a function of time a professional has spent in a professional field. The keynote speaker indicated that in other fields, innovative ideas most of the time are a result of hard work, superior skills and mastering all levels in a professional field. Would this apply to RE?
- New ideas versus new 'push-through ideas': how to distinguish between a truly new and original idea that would work, from ideas that will not work.
- 'Evil creativity': There are no studies about the usefulness and the mechanisms of creativity geared towards generating destructive ideas about what can possibly go wrong. Such ideas would be very useful to security requirements engineering and testing in order to improve a system´s quality (e.g. resilience, sustainability). Systems-of-systems, for which disaster recovery, crisis management, and safety are of prime importance, will greatly benefit from channeling the creative energy of testers in uncovering critical scenarios in which things could go wrong. The research on creativity in RE has so far focused on the phenomenon of generating and evaluating constructive ideas.

On the Sizes of Groups Using the Full and Optimized EPMcreate Creativity Enhancement Technique for Web Site Requirements Elicitation

Victoria Sakhnini[1], Luisa Mich[2], and Daniel M. Berry[1]

[1] Cheriton School of Computer Science, University of Waterloo
Waterloo, ON, N2L 3G1 Canada
vsakhnini@gmail.com, dberry@uwaterloo.ca

[2] Department of Industrial Engineering, University of Trento
I-38122 Trento, Italy
luisa.mich@unitn.it

Abstract. [Context] Creativity is often needed in requirements elicitation, i.e., generating ideas for requirements, and techniques to enhance creativity are believed to be useful. **[Objective]** How does the size of a group using POEPMcreate, an optimization of EPMcreate, affect the group's and each member of the group's effectiveness in generating requirement ideas? **[Method]** This paper describes an experiment in which groups of sizes one and two used POEPMcreate to generate ideas for requirements for enhancing a high school's public Web site. **[Results]** The data of this experiment combined with the data of two previous experiments involving groups of sizes two and four indicate that the size of a group using POEPMcreate does affect the number of raw and new requirement ideas generated by the group and by each member of the group. **[Conclusion]** The conclusion from the data is that the larger a group is the more raw and new requirement ideas it generates. However, there is some support for the conclusion that a group member generates more raw and new requirement ideas in a smaller group than in a larger group. This conclusion is supported by qualitative data gathered from a survey of professional business or requirements analysts.

1 Introduction

Many have observed the importance of creativity in requirements engineering, particularly for discovering and inventing requirements during elicitation of requirements for computer-based systems (CBSs) [e.g, 1–7], for those solving wicked problems, for those in highly competitive contexts, for those addressing critical business challenges, and for Web sites with requirements for high quality [e.g., 8–10].

Creativity is difficult to define, because it plays a role in technical innovation, teaching, business, the arts and sciences, and many other fields, and each field has its own definition. Creativity, in general, is the ability of an individual or a group to think of new and useful ideas [11]. Many techniques, e.g., brainstorming [12], Six Thinking Hats [13], and the Creative Pause Technique [14], have been developed to help people

be more creative. Some of these techniques have been applied to requirements engineering [15, 3], and some have also been subjected to experimental validation of their effectiveness [15–17]. A fuller discussion of creativity and of applying these techniques to requirements elicitation can be found elsewhere [18, 19, 17].

This paper investigates the use of an optimized and the full *EPMcreate (EPM Creative Requirements Engineering [A] TEchnique)* [19, 10] creativity enhancement techniques (CET) to help in generating ideas for requirements for Web sites. The optimization is called the Power-Only EPMcreate (POEPMcreate).

The feasibility of applying POEPMcreate and the full EPMcreate to help idea generation in requirements elicitation was established by earlier experiments [19, 10, 17]. The results of these experiments confirmed that:

1. EPMcreate helps generate more ideas and more new ideas for requirements than does brainstorming.
2. POEPMcreate helps generate more ideas and more new ideas for requirements than do EPMcreate and brainstorming.

These experiments exposed a number of issues to be explored in the future. These include the question that is taken as the research question of this paper:

> In each of EPMcreate and POEPMcreate, how does the number of members of an elicitation group affect the number of requirement ideas generated by the group and by each member?

The purpose of this paper is to begin to answer this question by conducting experiments in the context of eliciting requirements for a high school's Web site. In the rest of this paper, Section 2 describes the EPMcreate technique, including the POEPMcreate optimization. Section 3 describes the experiment, including its hypotheses and its steps. Section 4 gives the results of the experiment, analyzes them, and determines whether the hypotheses are supported. Section 5 discusses limitations of the results. Section 6 speculates about optimal group sizes, and describes the preliminary results of a survey conducted to obtain qualitative support for the results and speculation. Section 7 concludes the paper.

2 The Full and Optimized EPMcreate Techniques

Because EPMcreate is described fully *elsewhere* [19, 17] and space here is limited, the explanation of EPMcreate given here is abbreviated to that absolutely necessary to understand this paper.

2.1 Basic, Full EPMcreate

EPMcreate can be applied whenever ideas need to be generated, e.g., at any time that one might apply a CET, such as brainstorming. EPMcreate is by no means the only technique for identifying requirements; it is but one of many that can be used.

EPMcreate supports idea generation by focusing the search for ideas on only one logical combination of two stakeholders' viewpoints at a time. Sixteen such combinations are possible, each corresponding to one of the Boolean functions, fi for $0 \leq i \leq 15$, of two variables. If "Vn" means "Stakeholder SHn's Viewpoint", then these functions are $f0 = 0$, $f1 = V1 \wedge V2$, $f2 = V1 \wedge \neg V2$, $f3 = V1$, $f4 = \neg V1 \wedge V2$, $f5 = V2$, ..., $f8 = \neg V1 \wedge \neg V2$, ..., and $f15 = 1$. These sixteen functions are used to specify how the viewpoints of stakeholders SH1 and SH2 are combined in the sixteen steps of the EPMcreate procedure described in the next subsection, i.e., Step i combines the viewpoints of SH1 and SH2 according to the function fi.

2.2 EPMcreate in Practice

When a lead requirements analyst (leader) adopts EPMcreate as the CET for eliciting requirements for a CBS under consideration, she first chooses two kinds of stakeholders, SH1 and SH2, usually users of the CBS with different roles, as those whose viewpoints will be used to drive the application of EPMcreate. She may ask the CBS's analysts for assistance in this choice. She then convenes a group of these analysts. She shows the group only the Venn diagram of Figure 1 without the shading and the fis. In this diagram, the two ellipses represent two different stakeholders' viewpoints. Thus, for example, the intersection region represents the stakeholders' shared viewpoints.

The leader tells all convened analysts,

> *Today, we are going to generate requirement ideas for the CBS S in 16 idea generation steps. In all the steps, you will be pretending to think from the viewpoints of two particular stakeholders of S,* SH1 *and* SH2*.*
> - *In Step 0, you will blank out your minds ($f0 = 0$).*
> - *In Step 1, you will try to come up with ideas for problem solutions that are needed by both* **SH1 and SH2** *($f1 = V1 \wedge V2$).*
> - *In Step 2, you will try to come up with ideas for problem solutions that are needed by* **SH1 but not** *by* **SH2** *($f2 = V1 \wedge \neg V2$).*
> - *In Step 3, you will try to come up with ideas for problem solutions that are needed by* **SH1** *without concern as to whether they are needed by* SH2 *($f3 = V1$).*
> - *In Step 4, you will try to come up with ideas for problem solutions that are needed by* **SH2 but not** *by* **SH1** *($f4 = \neg V1 \wedge V2$).*
> - *In Step 5, you will try to come up with ideas for problem solutions that are needed by* **SH2** *without concern as to whether they are needed by* SH1 *($f5 = V2$).*
>
> ...
> - *In Step 8, you will try to come up with ideas for problem solutions that are needed* **neither** *by* **SH1 nor** *by* **SH2***, but are needed by other stakeholders ($f8 = \neg V1 \wedge \neg V2$).*
>
> ...
> - *In Step 15, you will try to come up with ideas for problem solutions without concern as to whether they are needed by either* SH1 *or* SH2 *($f15 = 1$).*

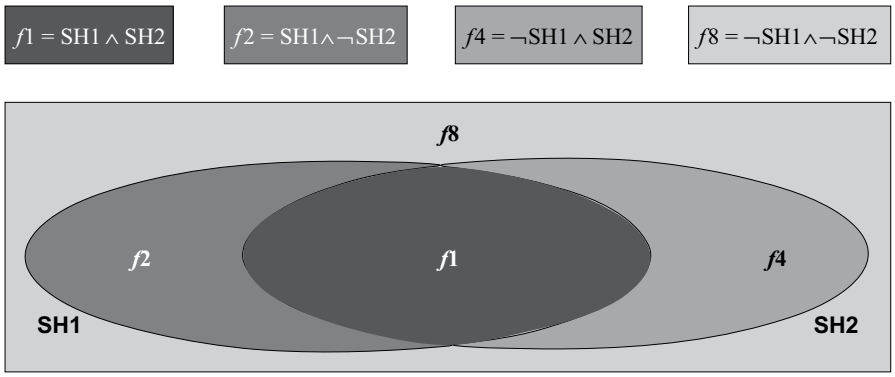

Fig. 1. The Four Steps of the Optimization and the Four Regions of the Venn Diagram

In the event that the leader believes that more than two stakeholders' viewpoints should be considered, she will convene more EPMcreate sessions, one for each pair of stakeholder viewpoints she believes to be useful.

2.3 Power Only EPMcreate

The optimization of EPMcreate that the research described in this paper studied is called the "Power-Only EPMcreate (POEPMcreate)", because it does only the four steps, described above, whose names include the powers of two, namely Step 1, Step 2, Step 4, and Step 8.

This optimization, which does only four of the sixteen original steps, was theorized, and later demonstrated [17], to be at least as effective as the full EPMcreate, because as illustrated by Figure 1, the Boolean function of each of the power-of-two steps corresponds to exactly one of the four regions of the Venn Diagram. Thus, the four power-of-two steps are sufficient to cover the entire space of potential ideas, and the other twelve steps just repeat the coverage.

3 Experimental Design and Planning

3.1 Background, Research Question, and Hypotheses

As mentioned, the feasibility of applying the full EPMcreate as a CET was established by previous experiments conducted by us [19, 17]. The effectiveness of POEPMcreate as a CET and as an *improvement* over EPMcreate was established by two previous experiments conducted by us [17]. The full description of the design, the conduct, and the results of these two experiments can be found elsewhere [17].

This paper describes a new, third experiment and a subsequent meta-analysis of the data of all three experiments combined. The purpose of this meta-analysis was specifically to address the question of whether the number of members of an elicitation group

using EPMcreate or POEPMcreate as a CET, affects the number of requirement ideas generated by the group and by each member. In fact, the sizes of the groups in the third experiment were chosen specifically to try, with the available subjects, to get data about group sizes not covered well or at all in the first two experiments.

When designing the third experiment and the meta-analysis, we could not predict with confidence any answer. It did seem reasonable that the number of ideas per group would be smaller with fewer members, but we really had no idea how the number of ideas per member would be affected by the number of members in a group. There were arguments supporting each choice of how the effect would go [12], but none of them seemed more compelling than the other. Therefore, we thought it is best to test only null hypotheses that address the research question:

H1 In each of EPMcreate and POEPMcreate, the number of members of an elicitation group has no effect on the quantity and quality of the requirement ideas generated by the group.

H2 In each of EPMcreate and POEPMcreate, the number of members of an elicitation group has no effect on the quantity and quality of the requirement ideas generated on average by each member of the group.

The new experiment's design and conduct were essentially identical to those of the previous experiments [17], including the choice of the CBS for which requirement ideas were to be generated. The sole differences between experiments were in the number of subjects, the number of groups, and the number of subjects per group. Since each of these differing numbers is an independent variable of the hypotheses, we are able to include the data of the previous experiments to address and test these two null hypotheses. The present experiment compared the requirement ideas generated for one CBS by four two-person groups and five individuals. We had no data points for individuals from the other experiments, and forming two-person groups got the maximum number of groups out of the remaining subjects. To simplify the rest of the paper, an individual is called "a one-person group". All groups used POEPMcreate and participated in the experiment for the same amount of time. Each group was to generate, using POEPMcreate, ideas for requirements for an improved version of one existing Web site, that of a Canadian high school with information directed to students, parents, teachers, and administrators [20].

3.2 Measuring the Effectiveness of a CET

The effectiveness of a POEPMcreate is measured by two numbers about the requirement ideas generated when using the CET,

1. the quantity of the generated requirement ideas, i.e., the raw number of requirement ideas generated, and
2. the quality of the generated requirement ideas, i.e, the number of high quality requirement ideas generated.

Counting raw requirement ideas is straightforward. The subjects wrote each idea on one line in Microsoft Word. About 90% of these ideas are in the form of one complete

sentence or a bullet item phrase describing a feature. Of the remaining 10% of these ideas, about 95% are at most two sentences.

The basis for evaluating the quality of a requirement idea is the notion that a creative requirement idea is both new and useful [16]. Therefore, as suggested by Mich et al. [19], the quality of each requirement idea was evaluated by classifying it into one of four categories:

1. new and realizable,
2. new and not realizable,
3. not new and not realizable, and
4. not new and realizable.

A requirement idea is considered new if it is not already implemented in the current Web site. "Realizable" includes several notions: (1) useful for at least one stakeholder, (2) technically implementable, and (3) socially and legally implementable, thus excluding privacy invading ideas.

To evaluate the quality of the requirement ideas in the experiment, each of two domain experts, namely the first and third authors of this paper, independently classified each idea into one of the four categories. The idea classifiers for this experiment are the ones who classified the ideas in the two previous experiments whose data are combined with those of the new experiment. As in the previous experiments, in order to reduce the chances that the authors' desired results might affect the quality evaluation, we merged the requirement ideas generated by all the groups into one file. We then sorted the ideas alphabetically to produce the list of ideas to be evaluated, making it impossible for any evaluator to see which group or individual, with its known CET, generated any idea being evaluated

As with the previous experiments, we considered requirement ideas in Categories 1 and 2, i.e., the new ideas, to be the high quality requirement ideas.

The two hypotheses H1 and H2 may be refined into eight different hypotheses, each one about the CET applied by a group, taking the number of requirement ideas produced either by the whole group or on average by a member of the group, an counting either raw or new requirement ideas, Each of the eight hypotheses is obtained by supplying one of two values for each of the three parameters in the sentence skeleton:

> H *CET measure kind*: The number of members of an elicitation group using [*CET*] has no effect on the [*measure*] of [*kind*] requirement ideas generated.

Here,
CET = "EPMcreate (E)" or "POEPMcreate (P)";
measure = "total number of ideas per group (T)" or "average number of ideas per group member (A)"; and
kind = "raw (R)" or "new (N)".

The eight hypotheses are, therefore:

HETR: The number of members of an elicitation group using **EPMcreate** has no effect on the **total number of ideas per group** of **raw** requirement ideas generated.

HETN: The number of members of an elicitation group using **EPMcreate** has no effect on the **total number of ideas per group** of **new** requirement ideas generated.

HPTR: The number of members of an elicitation group using **POEPMcreate** has no effect on the **total number of ideas per group** of **raw** requirement ideas generated.

HPTN: The number of members of an elicitation group using **POEPMcreate** has no effect on the **total number of ideas per group** of **new** requirement ideas generated.

HEAR: The number of members of an elicitation group using **EPMcreate** has no effect on the **average number of ideas per group member** of **raw** requirement ideas generated.

HEAN: The number of members of an elicitation group using **EPMcreate** has no effect on the **average number of ideas per group member** of **new** requirement ideas generated.

HPAR: The number of members of an elicitation group using **POEPMcreate** has no effect on the **average number of ideas per group member** of **raw** requirement ideas generated.

HPAN: The number of members of an elicitation group using **POEPMcreate** has no effect on the **average number of ideas per group member** of **new** requirement ideas generated.

Thus, each hypothesis whose name is of the form "H?T?" is a refinement of H1, and each hypothesis whose name is of the form "H?A?" is a refinement of H2.

3.3 Steps of the Experiment

Each experiment consisted of four steps. Steps 1 and 2 were done in one 50-minute meeting for each subject, and Steps 3 and 4 were done in one session with all groups, both two-person and one-person, in attendance.

The steps and their approximate times were:

Step 1: 20 minutes for each subject to fill out a general information form, to allow us to know his or her background: The form included questions about his or her age, gender, native language, computer science (CS) courses, qualifications related to CS, employment history in CS, and knowledge of the CETs: brainstorming, EPMcreate, and POEPMcreate,

Step 2: 30 minutes for each subject to take an adult version of Frank Williams's Creativity Assessment Packet [21], hereinafter called the *Williams test* to measure the subject's individual[3] creativity.

Step 3: 10 minutes for us to deliver to all groups an explanation about the experiment and POEPMcreate, the CET that they were to use. The explanation about POEPMcreate was basically the last two paragraphs of Section 2.2 of this paper, using only Function Steps 1, 2, 4, and 8.

[3] The phrase "individual creativity" is a technical term from the creativity assessment field that means *natural, unassisted, original creativity of the individual* and not just individual as opposed to group creativity.

Step 4: 120 minutes for each group to carry out its requirements elicitation session using POEPMcreate. Each group was provided with two computers: one with which to access the Web site that the group was to improve, and the other with which to write the requirement ideas generated by the group. The typical one-person group used only one of the computers to which it had access.

3.4 Assigning Subjects into Balanced Groups

To find subjects, we personally asked the University of Waterloo graduate students that were enrolled in a graduate-level advanced-topics in RE course to participate in the experiment for an honorarium of $20.00 (Canadian). Fifteen students replied, and of these, 13 ended up being subjects in the experiment.

These 13 subjects were distributed into 9 groups of sizes 2 and 1, as shown in Table 1, which shows also the demographic data gleaned from Steps 1 and 2. As in the previous experiments, we used these data about each subject from Steps 1 and 2 in order to create homogeneous groups with nearly equivalent spreads of CS knowledge, English fluency, work experience, and individual creativity. To make it *even possible* to form groups, we ignored gender and age as probably not relevant. As expected, none of the subjects had heard about any form of EPMcreate, even though all had heard about brainstorming. For these details about the groups in the other experiments, please consult the papers describing them [19, 17].

Table 1. Characteristics of Groups of The Experiment and Their Subjects

Group	# of subjects per group	# Males	# Females	# native in English	# not native in English	# taken ≥ 10 CS courses	# taken 3–5 CS courses	# worked professionally	# not worked professionally	Average age	Average Williams test score
G1	2	2	0	1	1	2	0	2	0	27.5	60.5
G2	2	1	1	1	1	2	0	2	0	26.5	76.5
G3	2	2	0	1	1	2	0	1	1	32.5	78
G4	2	2	0	1	1	2	0	2	0	26.5	86.5
G5	1	0	1	0	1	1	0	1	0	41	68
G6	1	1	0	0	1	1	0	1	0	25	72
G7	1	1	0	0	1	1	0	1	0	33	73
G8	1	0	1	0	1	1	0	0	1	21	79
G9	1	1	0	1	0	1	0	1	0	26	85

4 Experimental Results and Analysis

The new experiment was conducted in one session on 9 June 2010. The quantity and quality of the requirement ideas that were generated by the groups and individuals were

evaluated, as described in Section 3.2. This section reports the analysis of the combined data from all three experiments. Nevertheless, the tables and graphs make it clear from which experiment each datum comes.

Combining the data of multiple experiments, even ones essentially identical to each other, requires finding a data value about the subjects that is neither an independent nor a dependent variable of the experiments that establishes the similarity of the subjects and thus the comparability of the results. We use the Williams test score of each subject as this data value. The average Williams test score, out of 100, for the subjects in Experiment 1 was 70.81, in Experiment 2 was 73.44, and in Experiment 3 was 75.38. A single-factor analysis of variance (ANOVA) test shows that there is no significant difference between these averages. Consider also the one case of using data from two experiments for one independent variable, i.e., two-person groups doing POEPMcreate from Experiments 2 and 3. The Williams test scores of the members of the two-person groups in Experiment 2 ranged over 63 to 80.5 with an average of 72.9, while those of the members of the two-person groups in Experiment 3 ranged over 60.5 to 86.5 with an average of 75.2. Another ANOVA test shows that the difference between these averages is not significant. Therefore, it is legitimate to combine the date from the three experiments.

4.1 Results of the Evaluation of the Quantity of the Generated Requirement Ideas

As in previous experiments, the quantity of the requirement ideas generated by a group is simply the raw number of requirement ideas generated by the group. Table 2 shows for each group that participated in any of the three experiments: its experiment, identified by the experiment number 1 [17], 2 [17], or 3 [this paper]; its assigned CET; the number of persons in it; the number of raw requirement ideas it generated; and its average number of raw requirement ideas per person.

In the following results:

- Each correlation between an independent variable (IV) and a dependent variable (DV) was tested with a Pearson test. The result is described by an item of the form "**Correlation** (*IV*, *DV*) r-value, *assessment-of-the-strength-of-the-correlation*".
- Each difference between some dependent variable's value for two different group sizes was tested with a two-sample T-test for unequal variances. The result is described by an item of the form "(*phrase-describing-a-difference*) *assessment-of-the-significance-of-the-difference*, α-value, P-value".

For EPMcreate:

1. **Correlation** (a group's size, the number of requirement ideas generated by the group) $r = 0.79$, extra strongly positive
2. **Correlation** (a group's size, the average number of requirement ideas generated per person in the group) $r = 0.59$, strongly positive
3. (a member of a two-person group **generates more requirement ideas on average than** a member of a four-person group) significant, $\alpha = 0.055$, $P = 0.054$.

Table 2. Generated Ideas Per Group and Per Member in Three Experiments

Exp #	Assigned CET	Group Size	# Ideas Generated by Group	Average # Ideas per Member
1	EPMcreate	4	63	15.75
1	EPMcreate	4	60	15
2	EPMcreate	2	30	15
2	EPMcreate	2	35	17.5
2	EPMcreate	2	36	18
2	EPMcreate	2	40	20
1	POEPMcreate	4	74	18.5
1	POEPMcreate	4	76	19
2	POEPMcreate	2	40	20
2	POEPMcreate	2	42	21
2	POEPMcreate	2	45	22.5
2	POEPMcreate	2	63	31.5
3	POEPMcreate	2	66	33
3	POEPMcreate	2	30	15
3	POEPMcreate	2	90	45
3	POEPMcreate	2	67	33.5
3	POEPMcreate	1	27	27
3	POEPMcreate	1	30	30
3	POEPMcreate	1	18	18
3	POEPMcreate	1	18	18
3	POEPMcreate	1	27	27

Table 3. Generated New Ideas Per Group and Per Member in Three Experiments

Exp #	Assigned CET	Group Size	# New Ideas Generated by Group	Average # New Ideas per Member
1	EPMcreate	4	62	15.5
1	EPMcreate	4	56	14
2	EPMcreate	2	24	12
2	EPMcreate	2	26.5	13.25
2	EPMcreate	2	30	15
2	EPMcreate	2	21	10.5
1	POEPMcreate	4	70.5	17.625
1	POEPMcreate	4	70.5	17.625
2	POEPMcreate	2	32.5	16.25
2	POEPMcreate	2	32	16
2	POEPMcreate	2	36	18
2	POEPMcreate	2	51.5	25.75
3	POEPMcreate	2	46	23
3	POEPMcreate	2	20.5	10.25
3	POEPMcreate	2	57.5	28.75
3	POEPMcreate	2	68.5	34.25
3	POEPMcreate	1	15.5	15.5
3	POEPMcreate	1	19.5	19.5
3	POEPMcreate	1	29	29
3	POEPMcreate	1	18	18
3	POEPMcreate	1	17	17

For POEPMcreate:

4. **Correlation** (a group's size, the number of requirement ideas generated by the group) $r = 0.73$, very strongly positive
5. **Correlation** (a group's size, the average number of requirement ideas generated per person in the group) $r = 0.18$, weakly positive
6. (a member of a two-person group **generates more requirement ideas on average than** a member of a four-person group) significant, $\alpha = 0.05$, $P = 0.018$.
7. (**difference between** the number of requirement ideas generated on average by a member of a 2-person group **and** the number of requirement ideas generated on average by a member of a 1-person group) insignificant, $\alpha = 0.05$, $P = 0.2$.

4.2 Results of the Evaluation of the Quality of the Generated Requirement Ideas

As in previous experiments, the quality of the requirement ideas generated by a group is simply the number of new requirement ideas generated by the group. The classification

of the generated requirement ideas into the four ranks described in Section 3.2 was carried out as described in that section by the same two domain experts that did the classification in the two previous experiments [17]. The Pearson test shows that the correlation between the two experts' classifications was $r = 0.761$, a strongly positive correlation that is significant at the $\alpha = 0.01$ level. Therefore, it is reasonable to use each expert's and the average of the two experts' classifications and rankings of the quality of the generated requirement ideas.

In order to answer the research question with respect to the quality of the generated requirement ideas, we analyzed the combined data from all three experiments. Table 3 shows for each group that participated in any of the three experiments: its experiment, identified by the experiment number 1, 2, or 3; its assigned CET; the number of persons in it; the number of new requirement ideas it generated; and its average number of new requirement ideas per person.

For EPMcreate:

8. **Correlation** (a group's size, the number of new requirement ideas generated by the group) $r = 0.98$, super strongly positive
9. **Correlation** (a group's size, the average number of new requirement ideas generated per person in the group) $r = 0.57$, strongly positive
10. (**difference between** the number of new requirement ideas generated on average by a member of a 2-person group **and** the number of new requirement ideas generated on average by a member of a 4-person group) insignificant, $\alpha = 0.05$, $P = 0.082$.

For POEPMcreate:

11. **Correlation** (a group's size, the number of new requirement ideas generated by the group) $r = 0.81$, extra strongly positive
12. **Correlation** (a group's size, the average number of new requirement ideas generated per person in the group) $r = 0.1$, barely positive
13. (a member of a two-person group **generates more new requirement ideas on average than** a member of a four-person group) insignificant, $\alpha = 0.05$, $P = 0.1$, **but** significant, $\alpha = 0.11$, $P = 0.1$.
14. (**difference between** the number of new requirement ideas generated on average by a member of a 2-person group **and** the number of new requirement ideas generated on average by a member of a 1-person group) insignificant, $\alpha = 0.05$, $P = 0.32$.

4.3 Summary of Analysis

At the highest level, the results indicate that each refinement of the null hypothesis H1 should be rejected, albeit each with a different certainty. The reason for rejecting H1 is that, indeed, the number of members in a group *does* affect the number of raw and new ideas generated by the whole group: the larger of two groups generates more ideas. The results indicate also that each refinement of the null hypothesis H2 can be only barely rejected if at all. In fact, a few refinements cannot be rejected at all. Preventing a rejection is either that the numbers of raw or new ideas generated per member of two different sized groups are essentially the same or that the number of raw or new ideas generated per member of the larger of two groups is smaller than the number of raw or

new ideas generated per member of the smaller of the two groups. Now let's consider these hypotheses in detail.

The evaluation results 1, 8, 4, and 11 indicate that for each of EPMcreate (1, 8) and POEPMcreate (4, 11), the number of members in an elicitation group has a large effect on the numbers of raw (1, 4) and new (8, 11) requirement ideas generated by the group. The strength of the positive correlations between group size and the number of raw or new ideas ranges from .73 through .98 (4, 1, 11, 8). Thus, in order of increasing certainty, these results support the rejection of null hypotheses HPTR, HETR, HPTN, and HETN, each being a refinement of H1.

The evaluation results 2 and 9 indicate that for EPMcreate, the number of members in an elicitation group has a medium effect on the average numbers of raw (2) and new (9) requirement ideas generated by each member of the group. The strength of the positive correlations between group size and the average number of raw or new ideas generated per member ranges from .57 through .59 (9, 2). Thus, in order of slightly increasing, but almost identical certainty, these results support the rejection of null hypotheses HEAN and HEAR, each being a refinement of H2 to EPMcreate.

The evaluation results 5 and 12 indicate that for POEPMcreate, the number of members in an elicitation group hardly has an effect, if at all, on the the average numbers of raw (5) and new (12) requirement ideas generated by each member of the group. The strength of the hardly positive correlations between group size and the average number of raw or new ideas generated per member ranges from .01 through .18 (12, 5). Thus, in order of increasing certainty, these results can only barely support the rejection of null hypotheses HPAN and HPAR, each being a refinement of H2 to POEPMcreate. When broken down by the sizes of the groups compared, 4 & 2 and 2 & 1, these weaker correlations in the evaluation results 5 and 12 project out to more certain conclusions.

Specifically, for the comparison of groups of sizes 4 and 2, the evaluation result 6 says that it is significant at the $\alpha = 0.05$ or 0.055 level, that a member of a two-person group generates more raw requirement ideas on average than a member of a four-person group. Evaluation result 13 says that it is only with less significance, at the $\alpha = 0.11$ level, that a member of a two-person group generates more new requirement ideas on average than a member of a four-person group. Thus, there is some support for rejecting the null hypothesis HPAR and even less support for rejecting the null hypothesis HPAN when comparing members of groups of sizes 4 and 2.

For the comparison of groups of sizes 2 and 1, however, the evaluation result 7 says that the difference between the number of raw requirement ideas generated on average by a member of a one-person group and the number of raw requirement ideas generated on average by a member of a two-person group is *insignificant* at the $\alpha = 0.05$ level, and the evaluation result 14 says that the difference between the number of new requirement ideas generated on average by a member of a one-person group and the number of new requirement ideas generated on average by a member of a two-person group is *insignificant* at the $\alpha = 0.05$ level. Thus, the null hypotheses HPAR and HPAN cannot be rejected when comparing members of groups of sizes 2 and 1.

Ignoring temporarily the different strengths of the various null hypotheses that were rejected, let us consider the directions of these rejections. For each H1 null hypothesis, which compares, for each CET, the numbers of raw or new requirement ideas generated

by *entire* groups, the positive correlation between a group's size and the number of raw or new requirement ideas it generated says that for each CET, the larger a group's size, the more raw or new ideas it generates. This correlation is not surprising. For each H2 null hypothesis, which compares, for each CET, the numbers of of raw or new requirement ideas generated *on average by each member* of groups, the way that the null hypothesis is reject is, perhaps, surprising. A member of a two-person group generates on average *more* new or raw requirement ideas than a member of a four-person group does, and A member of a one-person group generates on average *approximately the same number of* new or raw requirement ideas that a member of a two-person group does. Thus, the power of a member seems to shrink with increasing group size, but the shrinkage in going from a one-person group to a two-person group is minimal. We say only "seems to" because the evaluation results leading to this conclusion are not very significant, if at all. This low significance is probably because there are only two data points for four-person groups while there are at least five data points for each of the one two-person and the one-person groups, and the spread of values for four-person groups is well within the spread of the same values for two-person and one-person groups. This shrinkage of a member's requirement idea generation power with increasing group size is evidence that group communication overhead, which grows quadratically with group size, detracts from its members' generating requirement ideas.

5 Threats to Validity and Future Work

Many of the possible threats to the validity of the conclusions to this experiment threatened the earlier experiments whose data have been combined with the data of this experiment. These threats were described and discussed thoroughly in the two papers about one of the previous experiments [17], and the reader is referred to there. These previously discussed threats include those of

- construct validity of
  - the ways to measure the quantity and quality of the generated ideas, and
  - the way to measure the individual creativity of each subject;
- external validity, including
  - the use of students as subjects instead of requirements elicitation or software development professionals,
  - the particular choice of the types of stakeholders whose viewpoints were used by EPMcreate and POEPMcreate sessions,
  - the single Web site as the CBS for which to generate requirement ideas, and
  - the small number of data points that, even in the presence of statistical significance, increases the probability that the positive observations were random false positives.

Internal validity, the new threat, is whether one can conclude the causal relationship that is being tested by the experiment, that the differences in groups size caused the observed differences in the quantity and quality of the requirement ideas generated. From the careful balancing of the groups and from what we have observed over several

experiments, we believe that the only factor that can account for the differences in the number of requirement ideas per CET is the sizes of the groups.

To address these threats to validity, we plan future experiments to get more data points and to do other experiments with different kinds of subjects, different sized groups, different stakeholder viewpoints, and different CBSs.

6 Postanalysis Speculation and Qualitative Triangulation

The definitive results indicate that when EPMcreate or POEPMcreate is used to help generate ideas for requirements elicitation, per group, a four-person group is more effective on average than a two-person group. However, *per group member*, a two-person group is more effective on average than a four-person group, and there is no significant difference between a one-person and a two-person group. Groups are traditionally thought to have synergy, by which the effect of a group is greater than the sum of the effects of its members [12]. These data suggest that synergy, if indeed it is present, is not very helpful. Perhaps, synergy is getting drowned out in a multi-person group by group communication, which grows quadratically with the number of persons.

To interpret the combined results of the three experiments, we designed and deployed in late August 2012 an online questionnaire that can be found at:
`https://docs.google.com/spreadsheet/viewform?formkey=dFI2UWx0MWJuRUdvQ1JNZnh1NFN0SGc6MQ`
The questionnaire's main goal was to learn what industrial practitioners knew about individual versus group requirements elicitation. Another goal was to learn the extent of the use of CETs in industrial requirements elicitation. This section gives a preliminary analysis of the data about individual versus group requirements elicitation from the 35 responses received by 22 October 2012. The answer to most questions involved choosing between "all", "most", "some", or "none" as an indication of the fraction of projects in which the statement of the question is true.

The answers to the question about the roles the respondent plays in his or her organization shows that many respondents are involved in more than one role, each of 34% of the respondents is a business or requirements analysts (BoRA) in all or most of his or her organization's projects, each of 14% is a software engineer (SWE), and each of 40% is a project manager (PM).

Requirements elicitation is described as an individual activity by a single BoRA working alone in all or most projects by 24% of the respondents, an individual activity by several BoRAs working separately in all or most projects by 9%, and a group activity in some or no projects by 44%. So, requirements elicitation is done by individuals in what appears to be a significant, non-majority, number of cases.

The usual number of BoRAs in a requirements elicitation group for all or most projects that use groups is given as 2 by 47% of the respondents, 3 by 37%, 4 by 15%, 5 by 0%, and greater than 5 by 11%. Thus, the smaller groups are favored.

When asked how to distribute an available 4 BoRAs to do a requirements elicitation task, 20% said that they would have the BoRAs work individually, 43% said that they would have the BoRAs work in 2 groups of 2, and 37% said that they would have the BoRAs work in 1 group of 4. Therefore, smaller groups seem to be chosen even when

more BoRAs are available for a task, and if more BoRAs are available, the extra ones would be used to make *other* groups rather than to beef up any one group.

It seems that BoRAs, PMs, and SWE in industry have noticed that smaller is better in forming groups for requirements elicitation, even without the benefit of controlled experiments. This observation suggests that the conclusions about EPMcreate and POEPMCreate that the experimental results weakly support may be correct and that more work needs to be done to strengthen the results.

7 Conclusions

The data from three experiments with essentially identical design and conduct are combined to draw conclusions that among EPMcreate or POEPMcreate elicitation groups,

1. increasing a group's size increases its overall effectiveness,
2. per group member, a two-person group is more effective on average than a four-person group, and
3. per group member, there is no significant difference between a one-person and a two-person group.

The lack of significance in Conclusion 3 together with the surprising Conclusion 2 leads us to speculate about optimal group size and the possibility that dividing the available EPMcreate or POEPMcreate practitioners into groups of two may be the best strategy. More work is needed to resolve this speculation.

For example, the data of Table 2 say that for POEPMcreate, a four-person group generates on average 75 raw requirement ideas, 18.75 per member, but a two-person group generates on average 55.38 raw requirement ideas, 27.69 per member. Thus, 2 two-person groups are expected to generate on average 110.76 raw requirement ideas, more than the 75, on average, generated by one four-person group. The question that needs to be answered is "how much of the average gain of 35.76 raw ideas is loss to duplication arising from the fact that the different two-person groups work independently?"

We are already conducting more experiments, primarily with four-person and two-person groups in order to equalize the number of groups of each size. We are letting the survey run longer in the hopes of collecting more responses, and we are analyzing the answers to all the questions and analyzing them more thoroughly to learn whatever we can from the questionnaires.

Acknowledgments

Each of Victoria Sakhnini's and Luisa Mich's work was supported in part by a Cheriton School of Computer Science addendum to the same Canadian NSERC–Scotia Bank Industrial Research Chair that is supporting Daniel Berry. Daniel Berry's work was supported in parts by a Canadian NSERC grant NSERC-RGPIN227055-00 and by a Canadian NSERC–Scotia Bank Industrial Research Chair NSERC-IRCPJ365473-05.

References

1. Gause, D., Weinberg, G.: Exploring Requirements: Quality Before Design. Dorset House, New York, NY, USA (1989)
2. Goguen, J.A.: Requirements engineering as the reconciliation of technical and social issues. In: Requirements Engineering: Social and Technical Issues, Academic Press (1994) 165–199
3. Maiden, N., Robertson, S., Gizikis, A.: Provoking creativity: Imagine what your requirements could be like. IEEE Software **21** (2004) 68–75
4. Hoffmann, O., Cropley, D., Cropley, A., Nguyen, L., Swatman, P.: Creativity, requirements and perspectives. Australasian J. Information Systems **13** (2005) 159–174
5. Maiden, N., Robertson, S., Robertson, J.: Creative requirements: Invention and its role in requirements engineering. In: Proceedings of the 28th International Conference on Software Engineering (ICSE). (2006) 1073–1074
6. Schlosser, C., Jones, S., Maiden, N.: Using a creativity workshop to generate requirements for an event database application. In: Proc. Int. Workshop Requirements Engineering: Foundation for Software Quality, REFSQ'08. LNCS 5025, Berlin, Germany, Springer (2008) 109–122
7. Nguyen, L., Shanks, G.: A framework for understanding creativity in requirements engineering. J. Information & Software Technology **51** (2009) 655–662
8. Rittel, H., Webber, M.: Dilemmas in a general theory of planning. Policy Sciences **4** (1973) 155–169
9. Rickards, T.: Creativity and the Management of Change. Blackwell, Oxford, UK (1999)
10. Mich, L., Berry, D.M., Franch, M.: Classifying web-application requirement ideas generated using creativity fostering techniques according to a quality model for web applications. In: Proc. 12th Int. Workshop Requirements Engineering: Foundation for Software Quality, REFSQ'06. (2006)
11. Runco, M.A.: Creativity: Theories and Themes: Research, Development, and Practice. Elsevier Academic Press, Burlington, MA, USA (2007)
12. Osborn, A.: Applied Imagination. Charles Scribner's, New York, NY, USA (1953)
13. de Bono, E.: Six Thinking Hats. Viking, London, UK (1985)
14. de Bono, E.: Serious Creativity: Using the Power of Lateral Thinking to Create New Ideas. Harper Collins, London, UK (1993)
15. Aurum, A., Martin, E.: Requirements elicitation using solo brainstorming. In: Proc. 3rd Australian Conf. on Requirements Engineering, Deakin University, Australia (1998) 29–37
16. Jones, S., Lynch, P., Maiden, N., Lindstaedt, S.: Use and influence of creative ideas and requirements for a work-integrated learning system. In: Proc. 16th IEEE International Requirements Engineering Conference, RE'08, IEEE Computer Society (2008) 289–294
17. Sakhnini, V., Mich, L., Berry, D.M.: The effectiveness of an optimized EPMcreate as a creativity enhancement technique for website requirements elicitation. Requirements Engineering Journal **17** (2012) 171–186
18. etourism Website: Online bibliographies, click on (1) creativity, (2) business creativity, (3) creativity techniques, or (4) brainstorming as a technique for software requirements elicitation (viewed April 2011) `http://etourism.economia.unitn.it/bibliographies/?locale=en`.
19. Mich, L., Anesi, C., Berry, D.M.: Applying a pragmatics-based creativity-fostering technique to requirements elicitation. Requirements Engineering J. **10** (2005) 262–274
20. Administrator: Sir John A MacDonald High School Web Site (Viewed 16–20 November 2009 and 7–12 March 2010) `http://sja.ednet.ns.ca/index.html`.
21. Williams, F., Taylor, C.W.: Instructional media and creativity. In: Proc. 6th Utah Creativity Research Conf., New York, NY, USA, Wiley (1966)

Requirements from the Void: Experiences with 1:10:100

Koen van Turnhout[γ], Stijn Hoppenbrouwers[γ], Paul Jacobs[γ], Jasper Jeurens[γ], Wina Smeenk[φ], René Bakker[γ]

[γ] HAN University of Applied Sciences.
Department of Information Technology, Communication and Media.
Ruitenberglaan 26, 6826 CC Arnhem,. The Netherlands
{Koen.vanTurnhout, Stijn.Hoppenbrouwers, Rene.Bakker}@han.nl

[φ] Independent Design Consultant
Zamenhofstraat 150 unit 328, Amsterdam, The Netherlands
wina@wien-s.nl

Abstract. In this paper we discuss our experiences with the 1:10:100 approach for organizing requirements elicitation in open innovation projects. 1:10:100 was originally developed to tackle the complexity of 'wicked' design problems, but also turns out to be a helpful means to organize requirements oriented pro ject conversations with heterogeneous groups of innovation partners. We use the 1:10:100 approach to shape project phasing. We discuss the approach, re port on our experiences using 1:10:100 for requirements elicitation in two ser vice design projects in the context of health care, and based on this present some practitioner's guidelines for using 1:10:100.

Keywords: 1:10:100, open innovation, requirements engineering, opportunity creation, service design

1 Introduction

Requirements Engineering (RE) comes in many different flavors, fitting as many different situations. The work presented here is rooted in the context of highly open design efforts, where both problem space and solution space are relatively unconstrained. In such situations, it is often problematic to get stakeholders to think 'out of the box' and explore novel opportunities rather than following familiar lines of thought, leading to (or even tracing back from) known solutions.

Maiden et al. distinguish between three veins in requirements engineering as a creative process: *Inspirationalist*, *Structuralist*, and *Situationalist* [1] (p62). Our current setting is very much Inspirationalist: "Focus[ing] on the interplay between consciousness and [sub]consciousness, opportunistic insight and associated breakthrough, leading to unexpected discoveries of new knowledge".

The 1:10:100 approach stems from design practice, not requirements engineering as such. According to Dorst [2] (p85) the approach was conceived at Stanford University as an educational device to bring 'hindsight' into design projects and to align 'research' activities with 'design' activities. Variants of the approach are in use as an educational device at Eindhoven University of Technology [3,4] and our own university [5,6]. Moreover, 1:10:100 is increasingly recognized as an effective approach for professional design projects [2,7]. Unfortunately, despite its growing momentum, the approach has so far remained rather badly documented (which we hope to remedy to a certain degree with this paper).

1:10:100 actively encourages 'discovery' in a design project; it is very flexible and open to growing insight. These characteristics make it useful for projects which are (1) opportunity rather than problem oriented, (2) that are open ended design projects and (3) where there is no clear idea of the preferable solution upfront [6]. In this paper we focus on describing the approach and illustrating its utility for organizing stakeholder discussions about requirements for innovative ideas in open innovation projects [8] where problem owners and innovation partners try to cooperate to find new avenues for innovation. We critically reflect on our use of the approach and we derive some guidelines for using 1:10:100 in the future.

2 The 1:10:100 Approach

The 1:10:100 approach is more a principle, with some best practices than a formal method. The core idea is to go through a complete design cycle three times within a project, with varying time spans [2]. Each design cycle consists of all phases of a traditional design project, including: research, specification, ideation, prototyping and evaluation. In each cycle, a new type of solution is pursued and a new concept is developed. The numbers in the name 1:10:100 refer to the length of each cycle. The first cycle (also called the pressure cooker) is done in a single day, the second in 10 days and the third in 100 days. In practice the actual number and length of iterations differs from project to project; the 1:10:100 ratio indicates planned upscaling of time and effort in three or more stages. At the end of each cycle an evaluation session is organized with all stakeholders to evaluate and to set focus for the next session. Various formal and informal design methods can be used within each cycle of the 1:10:100 approach. Over time we experimented with service design [11], community centered development [5,12] and integrative innovation [6].

The seven most important objectives of the 1:10:100 approach as we use it in open innovation projects are:

1. To create innovative concepts in open ended, opportunity oriented projects.
2. To facilitate a reflective design conversation with the innovation partners
3. To allow for early mistakes and discovery during the project
4. To align research and design activities in the project
5. To uncover, validate and balance a wide range of requirements
6. To jointly and gradually bring focus to the project with innovation partners
7. To create a common understanding with stakeholders for innovative solutions

In the remainder of this section we will discuss these goals in turn.

1. To create innovative concepts in open ended, opportunity oriented design projects

It is not easy to run a project that focusses on opportunity creation. Typically there is no straightforward problem that can be analyzed and solved. There are no existing users (in the narrow sense of the word) that can be studied and interviewed. It may be unclear what the product family of the solution will be, so it is hard to create a benchmark for the design. And finally, it is hard to plan the 'right' research and design activities, because it is unclear what the appropriate research and design questions are. This is why we speak of 'the void' in these projects. The difficulties mentioned can never be completely avoided but 1:10:100 is at least a partial remedy because of its strongly iterative nature and its focus on embracing early mistakes.

2. To facilitate a reflective design conversation with the innovation partners

In 1:10:100 the designers come up with new solution proposals in each design cycle, which are supported with tangible prototypes as illustration of the proposals. The idea is to fail gracefully in the first two stages (1 and 10) of the project. With the help of the tangible idea or concept directions from 1 and 10, discussions with stakeholders on the opportunities for change are more concrete and can guide discussions about underlying desires, needs and requirements. Since all partners are aware that solution directions are likely not to be pursued, there is ample room for a critical assessment of the strengths and weaknesses of a proposed solution. We refer to these meetings as quality review boards (QRB's). This iterative practice implements a joint form of 'reflective practice' as Schön defined it [9,10]. In his terminology, each cycle can be considered a design move, during the quality review board we evaluate the outcomes, we name the new priorities and issues and we frame the next cycle (Fig 1)

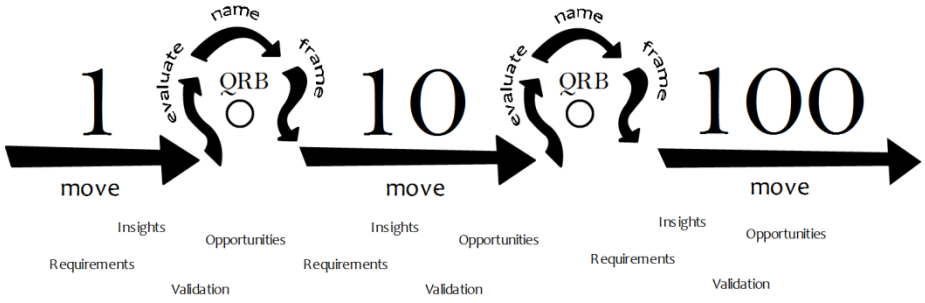

Fig. 1. The 1:10:100 as an organized reflective conversations consisting of a set of moves, evaluations, names and frames. Insights, requirements, validation of requirements and opportunities are harvested both in evaluation sessions (QRB's) and in the design cycles between those sessions.

3. *To allow for early mistakes and discovery during the project*

In 1:10:100 projects, designers are encouraged to make mistakes and to turn these failures into something valuable. By coming up with solutions which we plan to throw away, we act as problem seekers rather than problem avoiders. This gives freedom to the designers and the flexibility to try out multiple solution directions and to discover where the unanticipated opportunities are.

4. *To align research and design activities in the project*

In contrast to design projects which start with a formal analysis phase (a typically *structuralist* approach), the analysis in the '1' and '10' stages of 1:10:100 projects can be considered 'thin'. The outcome of the analysis does not give a complete picture and is certainly insufficient to be the backbone of a long project. However, the analysis is highly relevant to the design because it is done in close conjunction with the design activities. In a traditional project much of the rigorous research turns out to be irrelevant to the design in a later stage. In 1:10:100 the design focuses the research and vice versa, tackling the interdependency of the problem and solution domain that is characteristic of 'wicked problems' [13,14].

5. *To uncover, validate and balance a wide range of (possibly conflicting) requirements*

As designers explore new solution directions in each cycle, a wide range of solution proposals gets discussed during the project as a whole. So the challenge of the open innovation consortium is discussed from several angles, leading to a broad exploration of the problem space. As all proposed solutions address the original design brief there is also overlap between the requirements that have come up in the quality review boards. The requirements that are not tied to a particular solution get validated in the second (10) and third (100) cycle. However, we do not claim 1:10:100 is a silver bullet for balancing breadth and certainty in design projects: it takes considerable skill to uncover the 'right' requirements in a 'reliable way' within the three iterations of 1:10:100.

6. *To jointly and gradually bring focus to the project*

While goals 1-5 address the benefits of 1:10:100 as a design approach, in this paper we focus on its benefits for organizing conversations about requirements with a wide range of stakeholders in open innovation projects. Some of our partners are skeptical about putting effort in designing solution proposals that are thrown away later on, but in practice discussing how and why an early proposal is off the mark turns out to be an insightful exercise for everyone involved.

Problem owners can discuss their problem and think of requirements more easily when they discuss a concrete solution proposal (that they had not thought about before). Moreover, problem owners, who tend to be the more conservative partners in these projects, may open up during the early iterations as they see that their initial

concerns can be addressed in new ways. At the same time, partners (often the designers involved) that tend to be overly optimistic about the possible solutions in the beginning, may come to take the core requirements more seriously, become less naïve, and move more 'inside the box', during the project.

7. To create a common understanding with stakeholders for innovative solutions

A last important goal to use the 1:10:100 approach is to perform the reflective conversation mentioned under point (2) *jointly*, because choosing focus (framing) the design cycle builds involvement with, and acceptance of, the final solution direction with the partners. A 'cold' presentation of the innovative ideas that we develop during these projects would not work for many of the partners because the solution is too far away from the way they initially conceived it. Involving them in the reflective design conversations helps us to bring them along.

3 Lessons Learned in open innovation projects

3.1 Two case studies: family net and labor communication

In this section we discuss two bachelor graduation projects, focusing on opportunity finding for social media integration [5] and service design for health care institutions in the context of a transition to health care 2.0. Based on these projects, we perform a comparative case study [15]. In both projects the bachelor graduate acted as the chief designer, and a heterogeneous group of partners, including representatives from health institutions and our university, took part in an 'open innovation consortium'. The first case, referred to as *family net*, concerned collaboration between staff, seniors in care institutions and family and friends. The second case, *labor market*, focused on solving future staff shortage in care through novel approaches to the labor market. In the start of the projects, we visited the five care and nursing institutions which took part in the consortia for a discussion on the projects in considerable depth, to get an initial understanding of their expectations, explain the approach and secure their collaboration.

The first design cycle (1)

In both projects the goal of the first design cycle was to improve understanding of stakeholders, current working processes and user contexts, and to manage expectations. Interviews and literature scanning were used as a start. Techniques like mood boards, customer journey and application of existing design patterns were used to develop first designs. As design results, some scenarios were developed for different stakeholders. In *family net*, scenarios focused on a central information pillar in a care center. The pillar had a big red activity button that could be pushed to participate in activities. The scenarios in *labor market* were focused on using Twitter, Facebook and LinkedIn for announcing vacancies. In *labour market*, feedback on these designs was collected in separate meetings with all stakeholders. In *family net* this was done using a Quality Review Board (QRB) with all stakeholders.

The first design cycle in *family net* offered a sharp focus for users of the novel application: focus on family and friends, ignoring the difficulties of professional working processes and medical information sharing for the time being. Results of the first evaluation of the labor market project were disappointing. The problem owners mostly recognized the immediate usefulness of tools for distributing vacancies and could only comment on the perceived utility of the channels: they doubted whether LinkedIn would be appropriate. Other innovation partners criticized the straightforward design, but had little to go on to set or improve focus for the project.

The second design cycle (10)

In *family net*, the goal of the second design cycle was to gain an understanding of the role of family and friends in care and welfare processes, and in challenges and solutions offered by current services. This deeper understanding was obtained by means of participation in care centers, user surveys, an interview with a successful local community initiative and by joining a patient board meeting. Building on work concerning awareness systems [16] a mood app and shared experience and participation site was developed.

In *labor market*, the goal of the second design cycle was to elaborate on the role of staff as ambassadors for their care organization, to deepen the understanding of the role of HR in a novel situation and to explore the role of potential new staff members in this process. Basic techniques used were interviews with nursing staff, HR and potential employees. The designer used the costumer journey canvas to express his ideas [11].

Feedback was collected using a QRB in both projects. The most important resulting insight for *family net* was that the last design cycle should be directed at initiating participation by family and friends who are not a primary contact person (the large second ring of family and friends of a nursing home client). The most important result for *labor market* was that services should be developed for binding potential to a care institute, and services should be developed for immediate and future use of social media in staff recruitment.

The final design cycle (100)

In *family net*, the goal of the last design cycle was to identify triggers that would stimulate interest and participation of family and friends. An expert interview, cultural probes, co-creation and the application of psychological theory on remorse and guilt were used to gain contextual understanding. The result of this stage was a dynamic family portrait. The dynamic family portrait informs family and friends of events in daily life, shows a photo collage of people involved with a focus on people who visit the client. In addition, the application gently invites family and friends to participate by sending a new photo.

In *labor market*, two service blueprints were developed. In both blueprints, design principles were used that were derived from the first two cycles: transparency, authentic acquaintance building and staff ambassadors. The first design is a modest extension of current recruitment services: videos are used to show nursing staff activities

and the deep satisfaction of staff helping clients and working in teams. Some staff fulfill a role as ambassador of their institute. As an add-on on this concept, future services are directed at selecting and binding potential employees to the institution using a mix of services. The ambassador staff role is central in a mix of authentic experiences and regular interaction with the institution. In the future, vacancies are fulfilled by pooling a select group of potential staff, which has already had frequent contact with the institution through the services offered in an early stage of interest.

The resulting conceptual designs from 1:10:100 have led to prototypes that were welcomed by stakeholders involved. They showed both ideas for and a technical feasibility of novel concepts. Requirements engineering at the start of these projects would probably have resulted in quite abstract demands. 1:10:100 allows us to discuss much more concrete requirements rooted in scenarios, the importance of which is agreed on with the various partners. Thus, the prototypes are an invitation to do requirements engineering based on a sufficiently mature idea and on a common understanding of the novel application and its importance and relation to the original design brief.

3.2 Practitioners Guidelines

Against the background of our experiences with 1:10:100 in educational contexts, we compared the two open innovation projects described in section 3.1. We studied the actual proceedings of the projects, and their results, and set these against the goals described in section 2.1. The two cases were similar in terms of challenges, but different in the way 1:10:100 was used as an approach for organizing the requirements engineering process in the consortium. This allowed for a reasonably clean comparison between the two projects. In this section we will present 5 practitioner's guidelines that we obtained from these evaluations.

1. Inform partners and build trust around the process

At first, stakeholders who are not used to innovative design projects find it hard to see the value of 1:10:100 as an innovation approach. This plays out at two levels. First, stakeholders may not immediately see the value of the 1 and the 10 as joint discovery phases and consider it a weird idea to develop throw-away solutions. This can be remedied somewhat by a clear briefing and possibly by presenting inspiring examples from other projects. We used an initial client briefing to achieve this (see 3.1). In addition, it can help if the quality review boards are filled with a mixed group of people, some with experience in 1:10:100 and some with fresh minds. The second issue using 1:10:100 with inexperienced partners is that not all parties are immediately comfortable with critically assessing proposed solutions. The facilitator of the QRB needs to play an active role in provoking clients to express both negative and positive comments, and has to seriously address these by asking for underlying beliefs, values, problems and requirements. In the two cases we described, we intervened at points where the student designers did not know how to move on.

2. Start as a provocateur

Earlier on we described how in an 'ideal' 1:10:100 project, more conservative partners gradually open up across multiple QRB's, while more innovation oriented partners start to take requirements more seriously during the project, moving more 'into the box'. This process benefits from the designer taking risks in the 1 and 10 phases. For example, in *labor market* we experienced that the designer wanted to create an acceptable solution for the problem owners right away. This led to very conservative solution proposals, in turn resulting in a lack of feedback from the innovation partners. They were not challenged to consider the problem from novel angles, and the designer did not learn enough to come up with more fitting concepts later on. With this modest approach, the utility of 1:10:100 to take partners along in the exploration diminished; so the 1:10:100 goals of a joint reflective design conversation, joint focus finding and balanced requirements gathering suffered.

3. Create Common Ground and a Sense of Direction Together

It is not always possible or desirable to bring all partners together at a QRB and QRBs may not always be conclusive about the new directions to take. This can be problematic. In particular, a joint reflective conversation, joint focus and common understanding suffer if the innovation partners miss out on the perspectives of other partners. In *labor market* we avoided joint meetings because of difficulties with schedules. This led to email briefings, but they did not turn out to be effective in achieving joint focus. In the end the designer split his final design into two parts, one primarily for the innovation oriented partners and another primarily for the problem owners. We believe this is bad practice, therefore we stress the importance of organizing joint QRB's.

4. Design skill on the QRB

Not all QRB meetings proceed equally well. Usually we manage to involve experts and innovation partners in critiquing a proposed solution; in Schön's terminology [9,10] we manage to *evaluate*, but we do not always manage to set priorities (*name*) and focus the design (*frame*) adequately. It turns out that these are typical designer skills [2] and in those QRBs in which these aspects remain uncovered, we have to make up for them in the intervals that we leave between design cycles. This hinders the innovative concepts, reflective conversation and common understanding goals of 1:10:100.

5. Play the game for real

In a way 1:10:100 may feel as a game: there is something uncomfortable and silly about making and evaluating throw-away solutions. This can easily lead to an attitude in which early iterations are not taken seriously and evaluations are done sloppily. Some see the approach as 'the 100 is where it really happens –the rest is play'. The two projects we described in this paper hardly suffered from this attitude, but we often see it in student projects using 1:10:100. In some senses of the word 1:10:100 is indeed a game, but in other respects this is a counterproductive perception. If the early phases of 1:10:100 are 'just' play, then they are very serious play. For a designer in a 1:10:100 project, it is important to push herself to come up with a solution proposal which one believes in. For an innovation consortium too, it is important to go about the evaluations seriously. The lessons of the early stages are needed in order to be able to do things right in the 100 phase.

4 Conclusions

In this paper we described the 1:10:100 approach. We discussed how the approach can be beneficial for requirements engineering in open ended innovation projects with a heterogeneous group of partners. Also we shared our experiences in two open innovation projects with students acting as the service designers using 1:10:100 in various different ways. Based on our experiences we derived additional practitioner guidelines for using 1:10:100 in open innovation consortia.

We were surprised to learn that open innovation problems can be successfully addressed by students in bachelor programs of information systems and media design curricula of our university. The 1:10:100 approach supports us in developing an understanding by stakeholders of novel approaches to future opportunities. An unexpected benefit is that it facilitates our stakeholders, encouraging them to go along with us along the innovation path. The approach results in an understanding of future needs, by us and our stakeholders. Moreover, the 1:10:100 approaches allows us to bring a quick focus to the requirements engineering process for future applications, without compromising the innovation space of the designers.

In future research we plan to contrast the 1:10:100 approach to other requirements engineering approaches for radical innovation, to get a better understanding of the trade-off between costs and benefits of 1:10:100.

5 References

1. N. Maiden, S. Jones, K. Karlsen, R. Neill, K. Zachos, and A. Milne (2010): "Require ments Engineering as Creative Problem Solving: A Research Agenda for Idea Finding", in *Requirements Engineering, IEEE International Conference on*, Sydney, Australia, 2010, vol. 0, pp. 57 66.
2. Dorst, K. (2007). *Understanding design*. Gingko Pr Inc.

3. Hummels, Caroline, and Joep Frens (2008): "Designing for the unknown: A design process for the future generation of highly interactive systems and products.", *Proceedings Conference on EPDE*.
4. Tomico, O., V. O. Winthagen, and M. M. G. van Heist (2012):"Designing for, with or within: 1 st, 2 nd and 3 rd person points of view on designing for systems.", *Proceedings of the 7th Nordic Conference on Human Computer Interaction: Making Sense Through Design*. ACM.
5. Van Turnhout, K., Holwerda, R., Schuszler, P., Tijsma, L., & Bakker, R. (2011): "Interfacing Between Social Media, Business Processes and Users. A Design Case Study.", *Proceedings Chi Sparks 2011, Arnhem, Netherlands*
6. Van Turnhout, K., Leer, S., Ruis, E., Zaad, L., Bakker, R. (2011): "UX in the Wild: on Experience Blend & Embedded Media Design.", *Presented on The Web & Beyond Conference, 2012, Amsterdam, The Netherlands*. (http://bit.ly/SXXYg0)
7. See: http://www.luminis.eu/wp content/uploads/2011/11/factsheet 1 10 100.pdf
8. Chesbrough, H. W. (2003). *Open innovation: The new imperative for creating and profiting from technology*. Harvard Business Press.
9. Schön, D. A. (1999). *The reflective practitioner* (Vol. 1). Basic books.
10. Dorst, C. H. (1997). *Describing Design A comparison of paradigms*. Unpublished PhD Thesis, Delft University of Technology.
11. Stickdorn, M., & Schneider, J. (2010). *This is service design thinking*. Wiley.
12. Preece, J. (2000). *Online communities: Designing usability and supporting socialbilty*. John Wiley & Sons, Inc..
13. Lawson, B. (2006). *How designers think: the design process demystified*. Architectural press.
14. Rittel, H. W. J., & Webber, M. M. (1973). *Dilemmas in a general theory of planning*. Policy Sciences, 4 , 155 169.
15. Yin, R. K. (2008). *Case study research: Design and methods* (Vol. 5). Sage Publications, Incorporated.
16. Markopoulos, P., & Mackay, W. (Eds.). (2009). *Awareness systems: Advances in theory, methodology and design*. Springer.

The Even Darker Side of Creativity
Why Software Testers Should Check on Requirements

Hans Hartmann

OBJENTIS Software Integration GmbH
hans.hartmann@objentis.com

Abstract. Why employ testers to check on requirements? Are requirements not written and reviewed by business domain specialists? Testers approach their task with a special mindset, so a good tester will want to break the software—or to use *destructive creativity*—to find errors that others miss. This makes it advisable to begin testing before the software is even written. Supported by typical examples, this practitioner's report examines the basic reasoning behind the need for *destructive creativity*.

1 Introduction

In this paper we will try to show that it would be a good practice to include professional testers in the role of reviewers of software requirements. It is well understood that the importance of good requirements is essential for the success of any software project. However, when it comes to assuring the quality of the requirements, the industry as a whole generally does not utilize the capabilities of professional testers. This is what our (OBJENTIS) experience in performing integration, system and performance testing for diverse customers has revealed, for we have often found errors which could—or, indeed, should—have already been found during an earlier stage in the development process. Specifically, these were errors that could have been eliminated if the specifications had been addressed directly. In some cases, the specifications had been reviewed by business domain specialists, but their review did not find the type of errors that a professional tester surely would have found. We will trace this back to the unique way of thinking of testers—or what we more aptly refer to as the *destructive creativity* of testers. We will also address some of the economic implications of excluding testers, as we have encountered only three clients who actually do employ testers in the requirements review process. Importantly, this paper makes no attempt to describe how to write good requirements, as the emphasis is rather on how to detect insufficient requirements.

2 Insufficient Requirements

Two examples are used to show distinct types of failed requirements. The first type of insufficient requirement might typically be found in a contract and is based on a false assumption. The second is of the type commonly found in

software projects where a legacy software is replaced by a more modern version. The requirements are grouped in two categories: “all functionalities as they had been available in the previous version” and “additional requirements.”

2.1 A Wrong Requirement Causes Air Conditioning Breakdown

In 2010, several German ICE trains experienced delays due to exceedingly high temperatures in the carriages[1]. Measuring up to 50 °C the temperatures caused people to collapse and thus presented a serious health hazard. While Deutsche Bahn and the train manufacturer debated about who bore responsibility, it came to light that the trains' air conditioning systems were only designed to work if the outside temperature did not exceed 32 °C. We will therefore focus our attention on this specific requirement.

Although we do not know the exact wording, the requirement surely read something like this: Operation of the air conditioning system shall be guaranteed within the temperature range of minus something to plus something degrees. We know now, that the plus something was stated as 32 °C in the requirements. Another source reported that the air conditioning was designed to shut off completely if the outside temperature reached 38 °C. Be it as it may, we can say for certain that no requirement specified how the air conditioning should operate between 32 °C and 38 °C. Furthermore, nowhere was it defined how sufficient ventilation could be provided for the passengers under the given circumstances. Is it perhaps necessary to include such scenarios among the potential risks of traveling in summer? We think not, but this is indeed a valid example for our purposes, since it requires no specific knowledge of air conditioning systems or weather conditions in Germany apart from the ability to detect a false assumption[2].

2.2 Simple Requirement Deteriorates as Years Pass By

Legacy systems are modernized by porting code to another platform and changing the user handling. Frequently, the requirements are stated “as has always worked in the past.” However, specifications that were sufficient in content and wording in 1980 are likely no longer complete in 2013. In other words, could a business domain specialist in the year 1980 write a specification that accounts for potential problems occurring today?

3 The Software Development Process

Programs of reasonable size need an organized approach in building them, especially if several people are involved in the building process. The paradigm change

[1] `http://www.zeit.de/reisen/2010-07/deutsche-bahn-hitze-klimaanlage`

[2] namely, “Since Germany is not a tropical country, temperatures will not exceed 32 °C.”

from the 1980s until today has touched almost everything within the development process, but it has not really been able to improve the way requirements are defined except for the *agile development processes*[3]. Today, we have powerful requirements management tools available; and, in some cases, they are even connected to test management tools[4]. Nevertheless, we still have to answer two key questions:

1. Do we know our wishes?
2. Can we describe our wishes?

3.1 *Lastenheft* and *Pflichtenheft*

These two German terms - both closely related to "specification" - are used here to make a clear distinction in their meaning. *Lastenheft* (Engl.: product concept catalogue, product requirement document) is the specification book by which the contracting entity (the customer who orders the software to be made) has described its wishes. The producer of the software typically condenses the information in the specification book into a *Pflichtenheft* (Engl.: function specifications document).

We shall assume that there exists a customer who orders a specific software product to be built by a contractor. The customer will describe his wishes in a specification book[5], and will condense the information of the specification book in a functional specifications document. The producer of the software will extract the information of the specification book into a functional specifications document[6]. The German distinction between "Lastenheft" and "Pflichtenheft" ensures that the two different documents cannot be mixed up by mistake. In the case study, we will show the differences between a Lastenheft" and "Pflichtenheft", and the possible problems that arise from those differences[7].

4 Software Requirements

Our wishes and a description of our wishes are not the same. To describe wishes and transform them into unambiguous requirements, there exist books and guides e.g. [ALEX2002]. Knowing our wishes, on the other hand, has to do with psychology or *esoterical guidelines*.

[3] The waterfall model and V-model, both heavily relied on in the past, considered requirements as a fixed set of programming tasks. The requirements were checked as thoroughly as possible, since it was known that many errors already occur at this early stage. Nowadays, the prevailing opinion is that requirements will always change, and the new development processes (agile processes) favor the paradigm "embrace change"

[4] Since 2010, a new tool category is offered by the big players. It is called *Application Life Cycle Management*.

[5] product concept catalogue, product requirement document, "Lastenheft"

[6] the term functional specifications document is erroneous in itself, as it should contain not only functional specifications but also non-functional specifications.

[7] The example is derived from a real software project.

4.1 Knowing Our Wishes

Even expressing our wishes in informal terms requires knowing what we really want. Many fairy tales include a friendly character that is able to grant three wishes, yet the protagonist often fails to properly utilize this opportunity! Indeed, expressing a wish in a way that reflects one's true intentions can be difficult, and concessions must usually be made to meet the underlying goal—i.e., a successful career meant jeopardizing a happy marriage or an adventurous lifestyle meant neglecting one's children, etc. Thus, with any wish that we express, we also need to state what we want to avoid as part of achieving the goal. But even if we do so, most of us still neglect one key aspect. We may be able to define what we want and want to avoid in the future, but we usually do not consider special situations since most of us prefer not to think about them until they arise. As far as our personal wishes can be said to mirror software requirements, there are three important points to keep in mind:

1. What do we want? For software requirements, this is expressed by the business domain specialists.
2. What do we want to avoid? This is rarely mentioned in the requirements of the business domain specialists.
3. What could possibly happen that will render our original wish useless? This requires the unique way of thinking normally found among professional testers.

4.2 Describing Our Wishes

Describing our wish formally is the even more difficult task. In many books about requirement engineering, e.g., [ALEX2002], you will find rules about writing good requirements. Ian Alexander even states that it is necessary to create *misuse cases* [ALEX2003].

4.3 Working with Our Wishes

After carefully turning our wishes into requirements, we seek to implement the requirements in a software product that makes all of our wishes come true. We turn our focus to implementation, economic impact and risk management, but we do not speak about testing the requirements—thereby introducing a blind spot. Reviews and inspections are done by business domain specialists, and often everyone else other than testers. The latter become involved only at a much later time, during the software's development. In the following section, we will show why avoiding the use of testers early on means missing out on a great opportunity. In fact, only recently did we finally speak to a company that employs testers during the requirements phase[8].

[8] However, the quality manager conceded that it is done only for one specific line of the business.

4.4 Software Development Processes - A Closer Look

Software development processes have a great influence on the handling of requirements. They also treat testing in a very different way. In the *waterfall model* and in the *V-model* it is assumed that the actual testing is done at the end, while the Rational Unified Process RUP [KRU2004] calls for a much earlier starting date for testing. For their part, the currently favored *agile* processes place one form of testing, unit tests, even prior to development.

According to Crispin *et al.* [CRI2009], every team faces the requirements quandary and this also holds true for agile processes. Agile processes use their own method to formulate requirements—*user stories* which differ from fully fledged requirements. User stories are produced on the spot, quickly and they are definitely not complete.

In the same source we find the equation:

Requirement = *User Story* + Example/Coaching Test + Conversation

The term "test" in the above equation means a tester will be part of the development team; and this tester will ask the sort of questions that others normally neglect or simply do not think of.

The most important question coming from the tester is, "What's the worst thing that could happen?" Generally, every tester will ask this question, for it is a call to our *destructive creativity*. A tester has to think about the very worst scenarios, and must even invent them, because these scenarios never appear obvious. Bad scenarios that are imaginable can be contemplated by the developers, and they will program compensation for them. But how should we proceed if the worst case scenario has to be invented first? We have to look for people that are used to thinking in terms of worst case scenarios.

When using agile processes, testers are included in scrutinizing the user stories. At least two weeks after the user story has been defined, we will know if it has been implemented correctly and is sufficiently robust to stand the test once the application is running. We have brought testers very close to the requirements definition process. Indeed, not one requirement is worded without the contribution of a tester's thoughts.

However, many large projects seem to reject the agile development process.

4.5 Costs

To calculate the costs of an error, we can use the data estimation by Barry Boehm [BOEH1981] and the error distribution measured by Capers Jones [JONP2000] and combine these data. Barry Boehm [BOEH1981] states cost increases quite moderately with a factor of 75 in table 1. Using a common denominator of function points to make results comparable, Capers Jones [JONP2000] lists the distribution of errors depending on the development phase. (See Table 2.)

Table 1. Cost of Fixing Requirement Errors

Phase	Cost (Person–Hours)
Requirements	2
Design	5
Coding	15
Acceptance Test	50
Operation and Maintenance	150

Table 2. U.S. Averages in Terms of Defects per Function Point

Defect Origins	Defects per Function Point
Requirements	1
Design	1.25
Coding	1.75
Document	0.6
Bad Fixes	0.4

Taking Requirements and Analysis & Design together we end up with 45% of all errors in a program introduced in the initial phases of a program development. These errors can be found thereby saving a large amount of money in the entire course of program development. This fact has given reason for developing the development process *V-model*.

This paper would not have been written if it were generally accepted that professional testers are able to find some of the 45% of the *requirement errors* already in the requirement phase as opposed to only later during the validation process, in a system integration or acceptance test.

5 Testing

Testing computer software has been misinterpreted for a long time. **Let us be clear about what it is not:**

1. It is **not** proving that something works or does not work.
2. It is **not** checking if a program is error free.
3. It is **not** a job that everyone can do.
4. It is **not** something that can only be done in ultra-reliable environments.

Above all, software testing means finding errors and getting them fixed. Testing budgets are tight, which means we do not test for error-free programs. We test as long as our remaining testing costs are lower than the assumed risks if testing were halted. To test economically, we have to be organized and know the various ways of finding errors in the easiest and most efficient manner; and we must develop a feeling for potential weaknesses in the software.

There are two primary ways to do testing: formalized ways and exploring ways. Using formalized ways, we describe test scenarios, based on use cases, and

follow them. In exploring ways, we try to think of the software quality properties that we expect and head directly for them. The common expectation for finding them is the necessity of a foregoing search. A good tester's mindset has a built-in automated search function derived from experience and knowledge, as well as special skills that are taught to professional testers. Through 1999, only a few practical books had been written about software testing, e.g., the first, 1979, edition of [MYER2004] and the first, 1993, edition of [KAN1999].

In the last fourteen years, several helpful books have been published. We consider [KAN1999], [CRI2009] and [SPI2007] to be very good starting choices. In [KAN1999] at Page 363ff one finds an appendix of *common software errors*. In it, over 400 possible bugs are listed and described. A number of typical bugs should be added, as the complexity of software has since increased. Errors have even a bigger playground to smuggle themselves into. The mentioned appendix, or comparable listss of commonly expected errors, is a good starting point for comprehensive testing. However, we cannot rely entirely on such checklists. Imagine testing a new airplane. Each new plane is controlled against a checklist prior to takeoff. Nevertheless, the test pilot must invent and feel certain flight maneuvers in the air that he considers relevant for actual operation. We have our own checklist for testing requirements, and it can be found in Section 6. But we still keep our eyes open for errors that need to be categorized as OTHER.

What else do we do as testers?

1. We break software.
2. We organize disaster.
3. We test for quality.

5.1 Breaking Software

In the 1970s, we were a bit surprised to hear about techniques for injecting faults into hardware, where an electronic device would be probed at some internal spot. But instead of measuring voltage, current or a signal form, a special input would be forced in at this location, thus overriding the internal source of the original signal. We considered it quite harmful, for it could destroy some electronic components or even the entire board[9]. Today, we also use this technique in testing software. There is SWIFI (Software Implemented Fault Injection) which we can categorize in compile time injection or runtime injection. We mention it here for completeness[10]. (It is not relevant for checking requirements.) However, we use it to test the *robustness* of a software and that is a vital criterion as listed in the ISO9126 norm. ISO9126 describes the quality aspects of software.

[9] The linked page describes a related instrument. Signature analysis was not meant to destroy the tested unit. See `http://www.hpmemory.org/an/pdf/an\_222.pdf`.

[10] You may read about a practical aspect of this technique in `http://www.vtt.fi/inf/pdf/publications/2001/P448.pdf`.

5.2 Organizing Disaster

As testers, we try to produce disasters, whilst ensuring that we carry them out in environments that avoids harming production[11]. For example, in 1998 the development of a complex system of systems had to be tested for stability and recoverability. One of the tests included unplugging the power supply. Whilst today our programs have *autosave* functions and a sudden blackout would cause the loss of not more than the last five minutes' work, in 1998 a blackout meant the possible crash of several interconnected software systems, inconsistency of data bases and a long time to restart the whole system and get it running smoothly again. (The system was based on 31 semi-independent servers.) Testers account for these business production implications in the planning and enactment of disaster tests.

This example can easily be followed, but we must also consider other types of disasters—those that are not clearly visible before they occur. A key job of the tester is to anticipate them, or to use destructive creativity to invent such disasters. This ability of testers is also very useful when checking requirements. It will expand to think of voluntary and involuntary misuse of the program by the user[12].

5.3 Testing for Quality

A professional tester knows about the ISO/IEC 9126 standard[13] which discusses the six main quality aspects of software: functionality, reliability, usability, efficiency, maintainability and portability, with each property including further sub properties.

While these quality aspects can fill books of how to guarantee and test the compliance of a software, a strange phenomenon occurs when we consider requirements. In the transformation from *Lastenheft* to *Pflichtenheft*, the English translation states only functional requirements. This conceptual formulation suggests that business domain specialists are generally not expected to check on more than functionality. A tester, however, would check for usability, reliability and other topics of the ISO/IEC 9126 catalogue.

To merely request that a program functions well and then to leave the interpretation to the developer will not ensure that the requirements are complete. Rather, the provisions for the quality of a software program should already be prepared in the requirements phase. This is a function testers provide.

[11] There are cases where the production environment and the test environment are not separated completely. A tester wants to test accidental deletion of files and actually deletes data from the production environment.

[12] We know of a case where the accidental touch of the *Ctrl*-key together with a normal letter corrupted a database in such a way that it had to be reset completely. Although there exist very basic methods to avoid that the failure was caused by an error in the deployment process, where an old module of the early test times was accidentally exchanged for the actual module.

[13] `http://en.wikipedia.org/wiki/ISO/IEC\_9126` (Wikipedia used here, as the standard itself is not free of charge)

6 A Case Study

In 2011, our company was entrusted to check on two documents, a Lastenheft and a Pflichtenheft, in order to determine potential conflicts that could be detected before the actual software was programmed[14]. The review was based on the guiding rules of IEEE-1028 and IEEE-830, using two teams of testers. We used the guidelines for technical inspection. Several categories were monitored and the violations were counted. We targeted several categories:

1. Category Type 1: Understanding, Ambiguity, Implementation. Defects of these categories could have been avoided by observing guidelines for writing good specifications
2. Category Type 2: Contradiction, Missing, Completeness, Testability, Integration, Other (for special wordings in the documents). Defects of these categories need an overall assessment and special imagination. Which difficulties would we encounter during actual testing of the application?

Finding violations of the Category Type 2 requires the imagination of running real test cases. Test cases (or test scenarios) are designed in order to find errors. They try to describe situations that were not anticipated.

The statistical distribution of the findings is provided in Table 3. The case study shows that the number of errors that could be found by testers was greater than the number of errors that business domain specialists were able to avoid in the first place.

The customer confirmed to us that eight of ten vital findings of the highest severity range actually turned out to be the source of project delay and increased resource usage. *We do not claim that the data of this single investigation are representative of other requirements settings. However, we believe that the large percentage of violations in the* Missing *category is meaningful. It indicates that several important scenarios had not been considered in the requirements phase of the development process.*

Table 3. Error Distribution [%]

Category	Lastenheft	Lastenheft plus Pflichtenheft
Understanding	23	25
Ambiguity	7	0
Implementation	0	2
Missing	50	61
Completeness	4	0
Contradiction	11	8
Other	5	4

[14] The software was the customization of a real estate administration program in a SAP environment. Because of non-disclosure agreements, we cannot disclose any more detailed information about the case.

7 Summary

From the foregoing it can be concluded that professional testers, with their special way of thinking **destructively**, create advantages.

One can reasonably expect that some of the errors found by testers during the process of *validation* (e.g. during system test) could indeed be detected earlier. In turn, eliminating these errors in an earlier stage of development will reduce the cost of fixing them, as well as costs associated with additional deployment of the software.

Furthermore, by envisioning scenarios that could turn out to be bad, we may additionally detect scenarios that enable better maintenance of a product, or that reduce downtimes in cases of inevitable system failure. Importantly, asking what happens if something goes wrong will eventually develop into a standard routine procedure of checking for possible disasters.

Testers can improve neither the quality of the software (developers do that, after errors are reported) nor the requirements by themselves. Testers can, however, support a scrutinizing review of requirements in a very efficient manner, thus enabling the improvement of requirements.

Optimization of the various development processes has led to new *agile processes* and test driven development. As a result, starting with testing before a piece of software is written is nowadays a standard development method in the agile community. Undoubtedly, in non-agile development processes, too, performing professional testing while requirements are being written would encourage more efficient product and software development.

References

[ALEX2002] Alexander I., Stevens R.: Writing Better Requirements. New York: Addison Wesley , 2002.

[ALEX2003] Alexander I.: Misuse Cases: Use Cases with Hostile Intent. IEEE Software, vol. 20, no. 1, pp. 58-66, Jan.-Feb. 2003.

[BECK2000] Beck K.: Extreme Programming Expained: Embrace Change. Amsterdam: Addison Wesley, 2000.

[BOEH1981] Boehm B.: Software Engineering Economics. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[CRI2009] Crispin L., Gregory J.: Agile Testing. New York: Addison Wesley, 2009.

[JONP2000] Jones C.: Software Assessment, Benchmarks, and Best Practices. New York: Addison Wesley Longman, Inc., 2000.

[KAN1999] Kaner C., Falk J., Nguyen H. Q.: Testing Computer Software (2nd edition). New York: John Wiley & Sons, 1999.

[KRU2004] Kruchten P.: The Rational Unified Process (3rd edition). New York: Addison Wesley, 2004.

[MYER2004] Myers Glenford J.: The Art of Software Testing (2nd edition). New York: John Wiley & Sons, 2004.

[SCHW2004] Schwaber K.: Agile Project Management with Scrum. Sebastopol, CA: O'Reilly Media, 2004.

[SPI2007] Linz T., , Schaefer H., Spillner A.: Software Testing Foundations (2nd edition). Santa Barbara, Ca: Rocky Nook Inc, 2007.

3 International Workshop on Software Product Management (IWSPM)

Editors

Richard Berntsson Svensson
Lund University, Sweden, richard.berntsson_svensson@cs.lth.se

Inge van de Weerd
VU University Amsterdam, the Netherlands, i.vande.weerd@vu.nl

Krzysztof Wnuk
Lund University, Sweden, krzysztof.wnuk@cs.lth.se

Workshop Programme

7th International Workshop on Software Product Management (IWSPM 2013) 53
Richard Berntsson Svensson, Inge van de Weerd and Krzysztof Wnuk

Towards the identification of types of software product managers: tasks and situational factors 55
Katharina Peine, Andreas Helferich and Sixten Schockert

Towards Incorporating Sustainability while Taking Software Product Management Decisions 71
Birgit Penzenstadler, Mahvish Khurum and Kai Petersen

Mobile Game Software Product Management 87
David Callele

Assessing the Strategy Relevance of Delta Requirements A Bottom-up Feature Generation Approach 93
Gabriele Zorn-Pauli and Barbara Paech

Interactive Session: Cloud Computing Business Models – Trends and Implications for Software Product Managers 99
Hans-Bernd Kittlaus

7th International Workshop on Software Product Management (IWSPM 2013)

Richard Berntsson Svensson[1], Inge van de Weerd[2], and Krzysztof Wnuk[1]

[1]Lund University, Sweden
{richard.berntsson svensson,krzysztof.wnuk}@cs.lth.se
[2]VU University Amsterdam, the Netherlands, i.vande.weerd@vu.nl

Product success depends on skilled and competent product management. In essence, a product manager decides what functionality and quality a product should offer, to which customers, while minimizing the time-to-market and assuring a winning business case. The IWSPM workshop aims at bringing practitioners and research experts together for exchanging ideas, knowledge, and experience, and for setting a research agenda based on industry needs. Its main objectives are:

- Developing the software product management body of knowledge; identify challenges and future avenues for research relevant for both academia and industry.
- Strengthening software product management as a research field within the greater field of software engineering and business management.
- Providing software product managers and researchers a dedicated forum for exchanging ideas and best practices fostering industry-academia collaboration.

IWSPM 2013 included invited talks, paper presentations, and a panel on state of knowledge of software product management. The presented papers are included in the IWSPM 2013 proceedings.

IWSPM 2013 could not have been held without the help and support of a wide range of contributors. The advisory board consisted of Sjaak Brinkkemper, Christof Ebert, Tony Gorschek, and Samuel Fricker, who helped ensure continuity of the IWSPM workshop series. The program committee has contributed with timely and good quality reviews. Members of the program committee have been: Aybuke Aurum, Sjaak Brinkkemper, David Callele, Joerg Doerr, Christof Ebert, Remo Ferrari, Samuel Fricker, Paul Gruenbacher, Slinger Jansen, Lena Johansson, Marjo Kauppinen, Mahvish Khurum, Hans-Bernd Kittlaus, Nazim Madhavji, Andriy Miranskyy, Barbara Paech, Bjorn Regnell, Guenther Ruhe, Klaus Schmid, and Pasi Tyrväinen. We would like to thank the authors that submitted papers, the presenters who also opposed papers, and the participants of the workshop who contributed with valuable feedback by sharing their expertise, ideas, and opinions.

Richard Berntsson Svensson, Inge van de Weerd, and Krzysztof Wnuk
Program co-chairs IWSPM 2013

Towards the identification of types of software product managers: tasks and situational factors

Katharina Peine, Andreas Helferich, Sixten Schockert

Chair of Business Administration and Information Systems II (Business Software),
Universität Stuttgart, Keplerstr. 17, 70174 Stuttgart, Germany
{peine, helferich, schockert@wi.uni stuttgart.de}

Abstract. Constant change in their regulatory, economic and technical envi ronment forces organizations in many industries to adapt their software prod ucts to shifting demands. Many organizations have responded to this challenge by introducing the function of software product management. Setting up a suc cessful software product management depends on the organizational and tech nological conditions a company is confronted with. Consequently, one can as sume that different types of software product managers exist in practice. Aim of our research is to identify these types and the situational factors that determine the emergence of the types and the development of software product manage ment in an organization. To this end, we conducted a literature survey as well as an explorative empirical survey. Based upon these, we used a well established model from organization theory to postulate five types of software product managers. Additionally, we identified preliminary situational factors potentially influencing the embodiment of software product manager types.

Keywords: software product management, IT product management, functions, tasks, types of software product manager, typification, situational factors.

1 Introduction and methodological approach

Not only software companies or companies developing software-intensive systems, but virtually any company or organization providing services to customers using long-lived software infrastructures depend on the fit between their needs and the software product(s) fulfilling these needs.[1] To ensure this fit, many organizations have responded by introducing the position of the software product manager. Software product management guides and manages software products from inception to phase-out in order to maximize business value. This paper follows a functional perspective on software product management; i.e. the focus lies on the tasks assigned to software product management, not on the institutional naming of the position(s) performing these tasks.

[1] We define a software product as the result of a software development process, or the process itself, whose economic and technical potential allows targeted commercialization. [1]

Successful software product management needs to take into account the individual situation of an organization, e.g. its organizational, regulatory and technological conditions. A number of publications in the last years have examined the tasks assigned to software product management within an organization. [1, 3-6] All of these tasks cannot be fulfilled by one single position. Therefore, the fundamental hypothesis of this paper is that different types of software product managers exist in different organizations. We define a type as a group of people or things with similar qualities or functions that make them different from other groups. [1] However, up to today, there is only little support for the identification of types of software product managers and the underlying situational factors. Our paper wants to fill this gap.

We chose to proceed twofold. First, we conducted a preliminary, exploratory empirical study with over 230 participants working in the field of software product management (cf. section 2.1). This study reflects the real-life experience of practitioners in the field and demonstrates that software product managers in practice perform many diverse functions and tasks. We carried on with a systematic literature review on software product management, focusing on the tasks assigned to this functional area (cf. section 2.2). To set the results of the literature study on a solid basis, we searched for principles of organization theory to structure the findings in a theoretically founded way. [7-9] Together with Mintzberg's organizational configurations model [10] this serves as a basis for the identification of software product manager types (cf. section 3) and the description of their tasks (cf. section 4). Section 5 describes some first potential situational influence factors on software product management. Finally, in section 6, we provide an outlook on further research in the field of types of software product managers.

By drawing upon principles from organization theory and integrating a literature survey and an empirical study, this paper provides an important step towards the identification of different types of software product managers and the situational factors influencing the embodiment of these types in a real-world context. By this, the paper provides guidance for organizations dealing with software product management and expands the software product management body of knowledge. An overview of the procedure is given by figure 1.

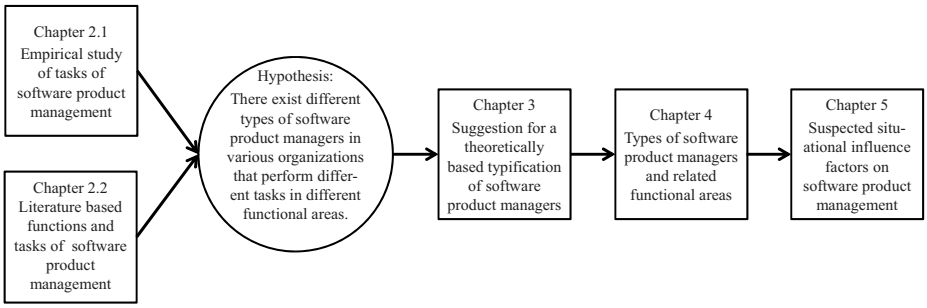

Fig. 1. Overview of the methodological procedure

2 Organizational functions and tasks of software product management

In order to increase the body of knowledge of software product management and to substantiate the previously presented hypothesis that there exist different types of software product managers in various organizations that perform different tasks in different functional areas, in the following the functions and related tasks of software product management will be examined more closely. On the one hand from a practical point of view, on the other hand based upon theory and literature.

2.1 Software product management tasks in practice

In the following, we present an empirical study on software product management tasks with over 230 participants working in the field of software product management. The participants come from software organizations that develop software systems as well as organizations that use software infrastructure in German-speaking countries and work for private companies or in public administration. They each participated in one of 28 software product management seminars in the years 2004 to 2012.[2] Following the definition of the functional view of software product management (cf. section 1), the job titles of the participants play a minor role.[3] Relevant is the competence to specify the main functions of software product management in their organization.

The study is mainly comprised of open questions about the major tasks of software product management. The answers were not expected to go into depth but focused on the most important aspects. The results are therefore exploratory and do not allow us to draw any conclusions on the individual situation of the organization that the participants come from. Therefore the study cannot give any hints on the participating organizations, e.g. differentiating in software vendors, vendors of software-intensive products, or internal IT organizations in non-IT industries. In addition, as a certain degree of saturation could be made out, it seems unlikely that a deeper survey, based on the tasks themselves as well as their order and distribution in terms of frequency of mention, would have resulted in further insights. Although some bias due to the preselection of the persons asked is possible (the participation in the seminars was charged), their distinct commitment to the topic ensures a good insight into the practice of software product management.

[2]The training seminars were held on behalf of the Management Circle AG, which is one of the most prestigious education agencies in Germany. Participants included knowledge holders from industry and government with several years of professional experience in software product management. The survey was regularly carried out.

[3]Even in an organization where no explicit position with job title “software product manager” exists, product management tasks can be performed, e.g. by a marketing manager or the managing director himself. On the other hand, a position owner carrying the job title “soft ware product manager” can perform tasks that are not attributed to the functional area of software product management. In this case, however, he has a different role, which is not considered in this paper.

The survey provided a total of 28 tasks, which represent software product management in its current business practice. Figure 2 shows the responses of the seminar participants in percent to the open kept question about the main tasks of the software product management in their organizations, sorted by frequency of mention.

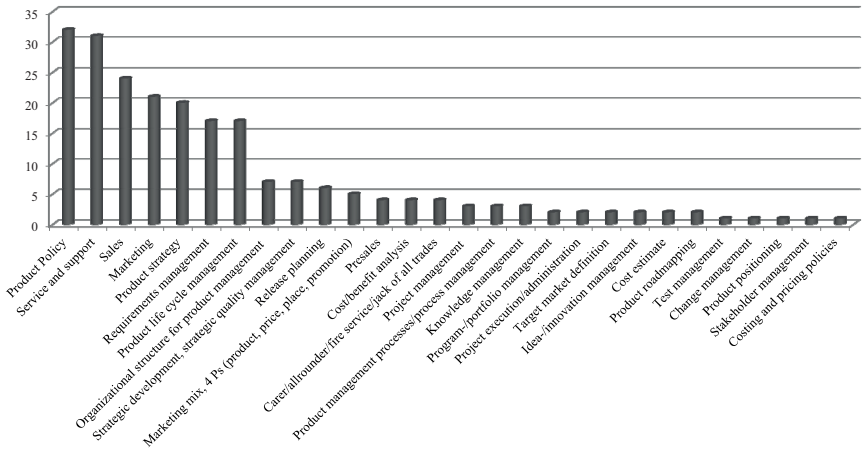

Fig. 2. Results of the explorative survey on the tasks of the software product management of seminars in the range of software product management

Software product managers in practice deal in first place with coordination tasks like product policy, and service resp. support actions at the interface between customers and developers. Together with the classical sales and marketing tasks this highlights the role of a software product manager as a multifunctional generalist, responsible for the optimal coordination of all product related tasks, both internally and externally. The software product manager takes care of the permanent flow of all product and market related information throughout the organization and deals with the market participants. Strategy decisions in combination with the management of the entire life cycle as well as the management of the requirements, both from the customers' and the developers' point of view, are also of particular concern.

The survey shows that software product managers in practice deal with very different tasks and occupy very different roles in their organizations. The priority of these functions and tasks varies greatly among the participants. It gives a first impression about software product management in practice as a discipline with established structures that shows very different characteristics. Because of the uncontrolled growth of tasks, a large degree of uncertainty and disorientation regarding the function exists. As a result, no consistent picture of the software product manager can be identified. Instead, the study suggests a situational outcome. The embracing hypothesis of this paper that there exist different types of software product managers in various organizations that perform different tasks in different functional areas can therefore be strengthened by this survey.

2.2 Software product management functions and related tasks in literature

Several publications on the tasks of software product management exist, but the authors do not define the various tasks in more detail nor do they provide a complete list of tasks (see e.g. [11-13]). However, the software product management body of knowledge (SPMBoK) of the International Software Product Management Association (ISPMA) structures 38 practices along functional areas and designates the core tasks of software product management as well as participation and orchestration tasks.[6] To supplement the SPMBoK and to obtain a comprehensive listing of functions and related tasks from scientific literature as well as the more practitioner-oriented literature, we conducted a systematic literature review on the topic.[4] In order to capture the entire subject area of software product management, the following search strings were used:

- (IT OR Software) AND (Produktmanagement OR Produktmanager)
- (IT OR software) AND ("product management" OR "product manager")

The analysis is based on a search in nine electronic, open access literature databases in economics and computer science/information technology. A special search in conference publications was abandoned because generally the conference proceedings are included in the databases (e.g. ICSOB in ELSEVIER, IEEE in IEEE Xplore, ECIS in Cite Seerx library, IWSPM in ACM DL etc.). A total of 984 German and English publications in the range of software product management were identified (see table 1), while table 2 lists the inclusion criteria for the systematic literature review.

After the identification of these publications a quantitative data analysis was conducted with the aim to weed out duplicates. In accordance with this, 781 publications remained. Based on the quantitative analysis, a qualitative analysis was executed: after critical reading of title, keywords and abstracts 57 sources were identified that deal with functions and tasks (exclusion criteria) of software product management. The list of these references can be found in the appendix. It demonstrates the current state of the art regarding the functions and tasks of software product management and provides a sound basis for further scientific and practical work.

[4] A systematic literature review is for the purpose of collection, selection, evaluation, interpre tation and presentation of available publications to differentiate a subject area from others, and to ensure the transparency of an analysis. Our analysis was based on guidelines for con ducting systematic literature reviews. [18 20] These approaches were integrated and adapted to the purpose of this analysis: Definition of the subject area, expiration of the analysis, lit erature search and search strategies, selection of relevant publications (inclusion criteria), quantitative and qualitative data analysis (exclusion criteria), data extraction, synthesis of publications and documentation. To ensure validity of the results as widely as possible, our review includes only peer reviewed publications, which we ascribe the necessary claim. Fur thermore, we proceeded in a structural way by using the principles from organization theory described below. The literature review took place between April 2012 and January 2013.

Table 1. Data bases and number of results per data base

Data base	Number of results
Association for Computing Machinery (ACM DL)	107
CiteSeerx library	232
Ebsco Host / Business Source Premier - BSP	113
ELSEVIER → Science Direct	191
FIZ Karlsruhe (io-port.net)	51
Google Scholar	128
Institute of Electrical and Elecronics Engineers (IEEE Xplore)	92
ISI Web of Science	33
SpringerLink	37
Total number of publications	**984**

Table 2. Inclusion criteria for results while the literature analysis

Inclusion criteria	Specification
(1) Type of publication	Book Paper
(2) Language	German English
(3) Keywords/search terms	- (IT OR Software) AND (Produktmanagement OR Produktmanager) - (IT OR software) AND ("product management" OR "product manager")
(4) Appearance of the keywords/search terms	Title Abstract Keywords

We analyzed the 57 sources from our systematic literature survey and extracted the functions and tasks of software product management. The examination of these references resulted in a very wide range of tasks of software product management. Subsequently, we matched the content of the functions and tasks and compared them qualitatively. In order to arrange the tasks more clearly, the tasks have been assigned to functions based on the organization theory principles of specialization and division of labor [7-9]: Functions within an organization can be defined as complexes of similar tasks. They can be divided into basic or core functions and cross divisional or service functions. The basic/core functions are directly necessary to fulfill the objective of an organization and can be further divided into planning or strategically-oriented functions and execution functions that belong to the process execution in the operational transformation system. With regard to contents, the tasks were structured and grouped using affinity diagrams. [14] The internal homogeneity of a region (in this case functional areas and tasks) requires elements within a group to be as similar as possible. External heterogeneity aims to achieve the strongest possible distinction of the regions (i.e. groups of functional areas and tasks) from each other. [15-17]

The analysis leads to 12 function areas with altogether 56 different tasks (see figure 3). In addition, the tasks are kept general, as they can be implemented individually by different organizations and can take different priorities depending on the situation as well as the organizational and technical structure. For further validation, we strive to conduct interviews on the topic with software product management experts (cf. section 6, further research).

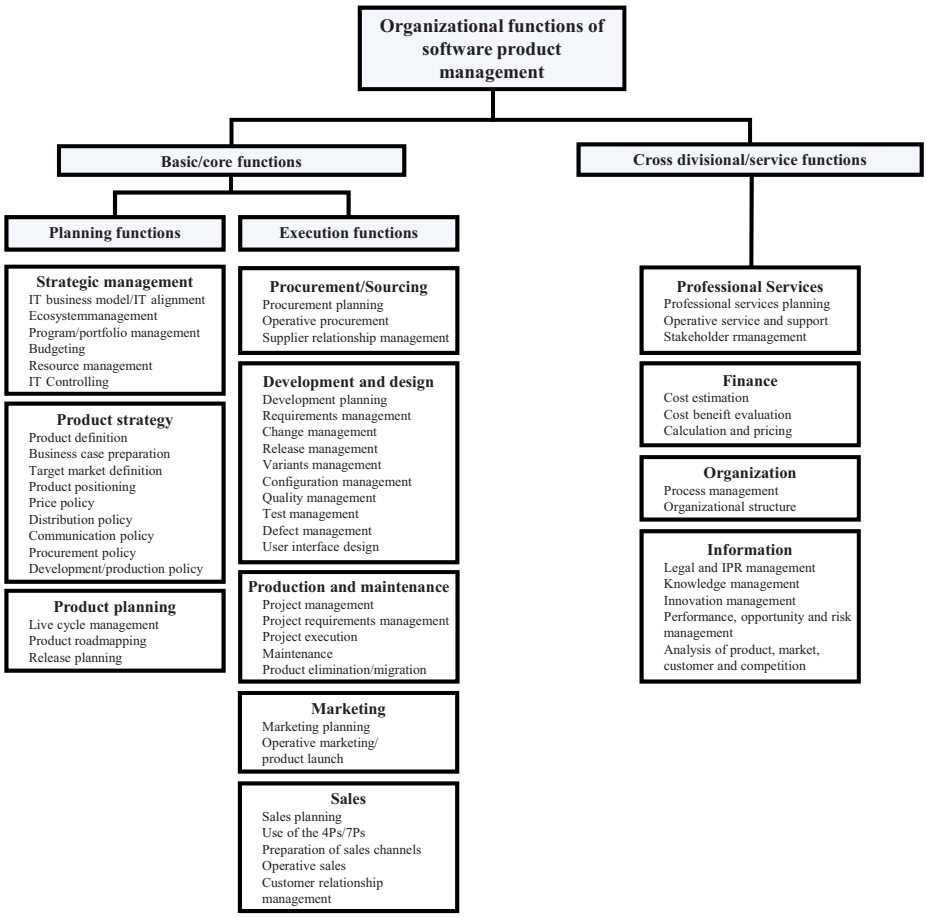

Fig. 3. Organizational functions of software product management

3 Suggestion for a theoretically founded typification of the software product manager

Software product management is still a relatively young scientific discipline. Only a small number of scientific publications exist, job descriptions and process models are missing. Since not all of the product management functions deduced from practice and literature can be performed by a single position holder, we are convinced that different types exist in practice. Division of labor and specialization are important organizational design principles. Thus, various types support the necessary allocation of tasks within an organization. Therefore, we derived different types of software product managers based on Mintzberg's organizational configuration model. [10]

A position (such as "software product manager") is the smallest, independently acting organizational unit that is equipped with responsibilities to perform a defined

complex of tasks. [2, 21] Thus, software product management as an organizational unit within the organizational structure describes an area of capabilities for an imaginary position holder. Mintzberg's organizational configuration model[5] suggests a position arrangement using the following classification of positions: instance, middle instance, execution, staff and service position. A *(middle) instance position* is located in the strategic apex or middle line and consists of top managers or managers arranged in a direct line of formal authority between people of strategic apex and operating core. It is equipped with professional performance powers, i.e. with decision-making and instruction powers. These are often planning, organization and control tasks in the organizations context. An *execution position* is situated in the operating core and instructed with the operational fulfillment of tasks. A *staff position* is located within the support stuff that provides indirect support to the rest of the organization. Therefore, the staff position assumes a supportive function for an instance position, and is in a suggestion role without instruction competencies. A *service position* is arranged in the techno structure that consists of those analysts, outside of the formal "line" structure, who apply analytic techniques to the design and maintenance of the structure and to the adaptation of the organization to its environment.

Following this approach and based on Mintzberg's model, we developed a theoretically founded typification of software product managers (illustrated in Figure 4).

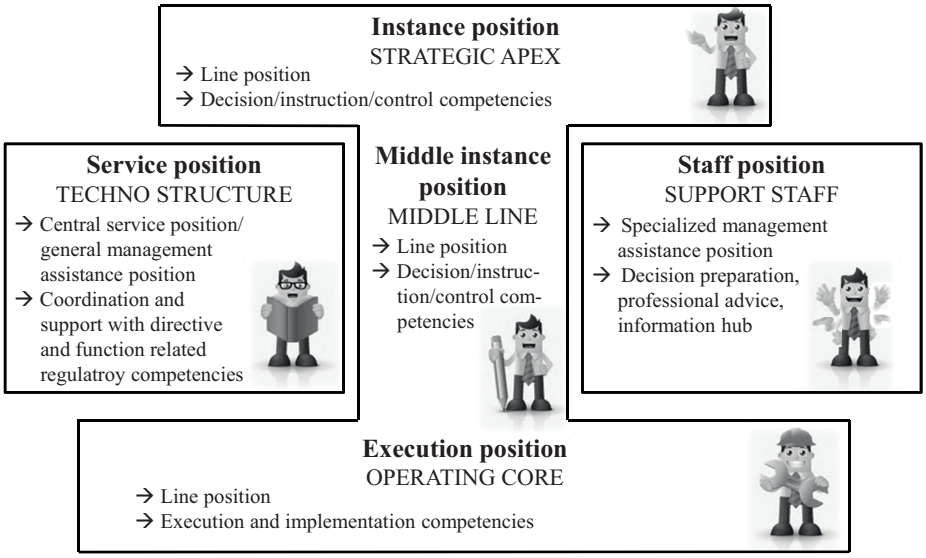

Fig. 4. Suggestion for a theoretically founded typification of the software product manager

The software product manager in an *instance position* aligns a variety of tasks with each other, acts as an information hub, brings together all the participating actors, matches them, delegates tasks and ensures that the responsible authorities have all

[5] Mintzberg's standard model is considered to be a "classic of organization theory" and still serves as the basis for numerous research approaches. [22]

important information at any time. He has a long term view and knows (or even decides) about strategic issues. Entrusted with an *execution position* the software product manager inheres both, the capability and the responsibility for the task with which he is entrusted. The fulfillment of the task is difficult when the software product manager is responsible for the result, but does not have the necessary influence, in the sense of decision-making power, to push through decisions. Therefore, it is important that competence and responsibility fall together (principle of congruence). [23-24] In the *staff position* the responsibility is not with the software product manager. He provides a basis for decision-making of higher authorities, supports one or more instance positions, and takes an advising part. The software product manager in a *service position* assists other positions in providing services and fulfills supportive tasks.

4 Types of software product managers and related functional areas

Practical experience and literature review reflect that no consensus on software product management tasks exists and that a general task description is not possible. Tasks depend on the organization and the environmental conditions. In the previous section we derived five types of software product managers that will now be assigned to the organizational functions of software product management (cf. section 2.2). In practice there may be overlaps of the types and tasks, as well as one type of product manager must not necessarily perform all the tasks that are assigned to him. However, the presented types constitute a basis typification. Figure 5 shows the derived types in relation to the functional areas.

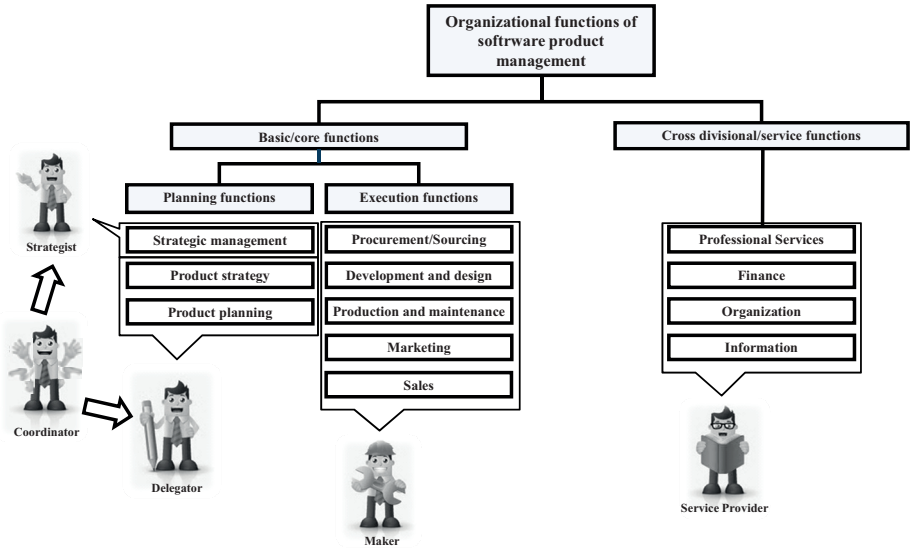

Fig. 5. Types of software product managers and related functional areas

The *Maker* as a line position in the operating core within the value structure of an organization has execution and implementation competencies. He predominantly he performs tasks in the execution function areas of procurement/sourcing, development and design, production and maintenance, marketing and sales. Consequently the focus of the Maker is not the product itself, but the actual fulfillment of tasks. He has no long-term view on things. He performs tasks that are assigned to him by the other types of software product managers or other position holders within the organization. Ideally he has the responsibility for the task with which he is entrusted. Because of this lack of authority he is sometimes not even perceived as a software product manager.

The *Service Provider* holds the service position in the techno structure within the cross divisional/service function areas professional services, finance, organization and information. Thus he is fixed to the product itself and has a medium-term view on to all transactions that have to do with the product. He knows about the direction of strategic management, product strategy and product planning and therefore is informed about planning issues. Above all, he supports the related instances with his expertise, professional skills and specialized knowledge. He assists the other positions in providing services and adapts the product to the organization's environment.

The *Coordinator* is assigned to one or more instances of the strategic apex or middle line. Hence, he is settled in the functional area "planning". He is located in the support staff, where controlling and coordinating tasks are ascribed to him. The Coordinator is acting as a product-oriented interface and can be seen as a control center. This solves a lot of problems, e.g. interface problems due to lack of agreement between different stakeholders and even competition between the involved departments or problems with communication between the different stakeholders in general. If there is a position within the organization that is explicitly responsible for coordination, problems of unclear or unspecified coordination processes can be addressed. Some product managers even describe their job profile with quite unusual phrases like caretaker, all-rounder, handy man or jack of all trades. They perceive themselves as troubleshooters or rapid deployment unit in the sense of a fire service. The Coordinator can therefore be regarded as a multifunctional generalist and central communication platform that coordinates and controls product related decisions. He acts supportive and provides a basis for decision-making for an instance position.

The *Strategist* and *Delegator* are both instance positions in the strategic apex respectively in the middle line within the functional area "planning". They fulfill decision-making tasks. The Strategist acts in the function area "strategic management", whereas the Delegator deals with the areas of product strategy and product planning. The focus of these instance positions is on business development. This implies a long-term view on things. The Strategist performs almost exclusively strategic tasks, has budgetary control and aligns the product to the context of the organization so that the goals of the organization correlate with the product strategy. The Delegator deals with software product management itself. Both of these instance positions can be supported by a Coordinator.

5 Situational influence factors on software product management

In order to provide concrete design recommendations, knowing the action alternatives in form of possible software product manager types is not sufficient. So far, the types of software product managers connect their tasks and responsibilities. However, the hypothesis introduced in section 1 states that the implementation of a software product manager also depends on other situational factors. It is likely that the different types are not only deduced from their tasks, but also from other conditions. The performance of the task execution is at least dependent upon the goals an organization is heading for and the environmental conditions it is confronted with. Moreover, company-specific factors like company size, organizational form and product characteristics have to be considered. Groundwork in this field has already been accomplished by the study of Bekkers et.al. [25] They emphasize the influence of situational factors on software product management practice but complain that research done so far has mainly focused on singular tasks. Furthermore, Clarke and O`Connor have developed a framework that demonstrates situational factors influencing the software product management and divide them into categories. [26]

On this basis, we try to derive situational factors which determine the types of software product managers. Josten takes the basic components of organizational situational factors and adapts them to product management. [27] We aim to adapt these basic components (organizational structure, dimensions of the internal and external situation and the actual and cognitive behavior of the organization's members and their effects) to possible components of software product management. An outlook on potential situational factors and their effects on typification is shown in Figure 6. Some potential effects of situational influence factors include:

The smaller the organization, the more likely it is that the business director/CEO assumes many (strategic) tasks by himself, while he delegates execution tasks to a Maker.

The higher the level of decision delegation in an organization, the more likely the software product manager is incorporated as a decision-making authority in the business organization (Strategist or Delegator).

The greater the department of software product management itself, the more likely a Strategist or Delegator is needed that heads over other positions such as Coordinator, Service Provider or/and Maker.

The higher the standardization in the software product management, the more extensive is the specification of tasks. Tasks may be performed by an execution position such as Maker.

The more heterogeneous the product/service portfolio is set up, the more likely a Coordinator or even Delegator/Strategist is necessary. Otherwise, to get a single, clearly defined product developed sustainably it is under certain circumstances sufficient if the requirements management is done properly by a Maker.

The more (various) internal or external customers or suppliers exist, the more coordination is necessary by Service Provider/Coordinator.

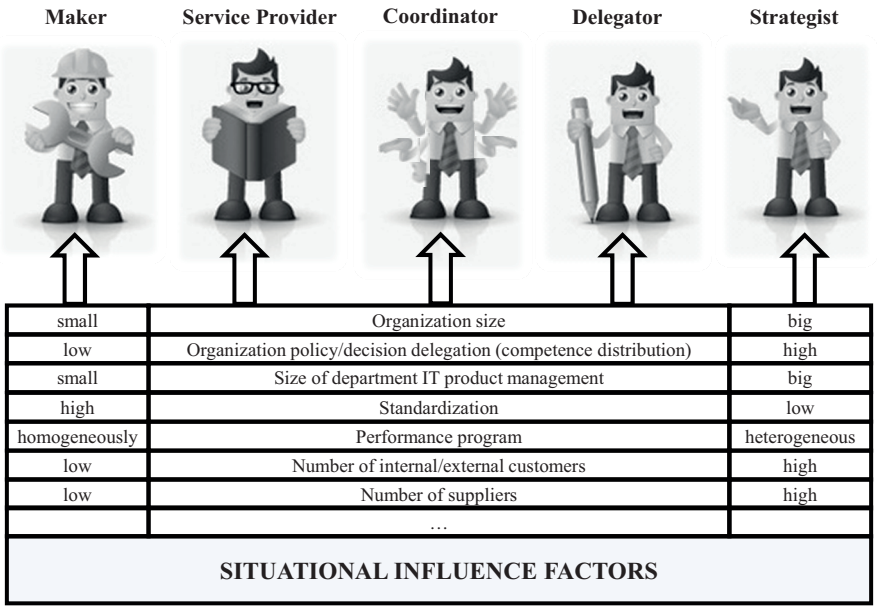

Fig. 6. Potential situational influence factors and their effects on software product management

6 Conclusion and outlook on further research

The hypothesis introduced in section 1, that there exist different types of software product managers in various organizations that perform different tasks in different functional areas, could be substantiated by empirical and theoretical evidence. The explorative empirical study showed a great variety of software product management practices. By dividing the tasks of software product management into functions, a foundation for the typification could be created. Based on Mintzberg's organizational configurations model, types of software product managers could be derived and assigned to functional organization areas. Furthermore, a first proposal for situational factors influencing software product management could be made. Further research is necessary to elaborate the relevant design parameters and to align them to coherent software product management configurations.

The organizational functions, related tasks and types of software product managers add value to the software product management body of knowledge by presenting the range of possible tasks systematically. Software organizations can profit by better defined job descriptions and guidelines for the segmentation of tasks and responsibilities. Education institutions can benefit because software product management curricula can be better adjusted to the requirements of practice. In a next step we intent to adapt knowledge from current industry standards in the range of software product management as well as carry out an analysis of job offerings for software product managers. To further validate and expand our results we strive for in-depth expert interviews. The goal is a well-founded, situational model of software product man-

agement types including design recommendations for setting up software product management in practice.

References

1. Herzwurm, G., Pietsch, W.: Management von IT Produkten. dpunkt, Heidelberg (2009)
2. Kosiol, E.: Organisation der Unternehmung. Gabler, Wiesbaden (1962)
3. Kittlaus, H.B., Clough, P.: Software Product Management and Pricing Key Success Fac tors for Software Organizations. Springer, Berlin and Heidelberg (2009)
4. Van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L.: On the Creation of a Reference Framework for Software Product Management: Validation and Tool Support. First International Workshop on Software Product Management (IWSPM '06), pp. 3 12, Minneapolis (2006)
5. Peine, K., Helferich, A., Schockert, S.: Nachhaltige Anwendungssysteme dank IT Produkt management. In: Brandt Pook, H., Fleer, A., Spitta, T., Wattenberg, M. (eds.): GI Edition Nachhaltiges Softwaremanagement, LNes in Informatics,pp. 36 49, Bielefeld (2012)
6. Fricker, S.: Software Product Management. In: Maedche, A.,Botzenhardt, A., Neer, L. (eds.): Software for People Fundamentals, Trends and Best Practices. Springer, pp. 53 82, Heidelberg u.a. (2012)
7. Robbins, S.: Organization Theory the structure and design of organizations. Prentice Hall, Englewood Cliffs (1983)
8. Daft, R., Murphy, J., Willmott, H.: Organization Theory and Desing. Cengage Learning EMEA, Cheriton House u.a. (2010)
9. Witte, H.: Allgemeine Betriebswirtschaftslehre: Lebensphasen des Unternehmens und be triebliche Funktionen. 2nd ed., Oldenbourg Wissenschaftsverlag, München (2007)
10. Mintzberg, H.: The Structuring of Organizations. Prentice Hall, Hemel Hempstead/Engle wood Cliffs (1979)
11. Condon, D.: Software Product Management. Aspatore Books, USA (2002)
12. Dver, A.: Software Product Management Essentials. Meghan Kiffer Press, Tampa (2003)
13. Sneed, H., Hasitschka, M., Teichmann, M.: Software Produktmanagement Wartung und Weiterentwicklung bestehender Anwendungssysteme. dpunkt.verlag, Heidelberg (2005)
14. Martin, B., Hanington, B.: Universal Methods of Design: 100 Ways to Research Complex Problems, Develop Innovative Ideas, Design Effective Solutions. Rockp. Publ. USA (2012)
15. Bailey, K.: Typologies and Taxonomies: An Introduction to Classification Techniques. Sage, Thousand Oaks u.a. (1994)
16. Guba, E.: Toward a Methodology of Naturalistic Inquiry in Educational Evaluation. In: CSEMonography Series in EvaluationNo. 8,University of California, Los Angeles (1978)
17. Kelle, U., Kluge, S.: Vom Einzelfall zum Typus: Fallvergleich und Fallkontrastierung in der qualitativen Sozialforschung. VS Verlag, Wiesbaden (2010)
18. Kitchenham, B., Charters, S.: Guidelines for performing Systematic Literature Reviews in Software Engineering. Keele University and Durham University, Technical Joint Report EBSE200701(2007)
19. VomBrocke, J., Simons, A., Niehaves, B., Riemer, K., Plattfaut, R., Cleven, A.: Recon structing the giant: on the importance of rigour in documenting the literature search process. In: 17th European Conference on Information Systems, vol. 3, issue 15, pp. 1 13 (2009)
20. Maglyas, A., Nikula, U., Smolander, K.: What Do We Know about Software Product Man agement? A Systematic Mapping Study. 5th International Workshop on Software Product Management (IWSPM'11), Trento, Italy, pp. 26 35 (2011)

21. Bea, F.X., Göbel, E.: Organisation. Lucius & Lucius, Stuttgart (2006)
22. Nicolai, A.; Vollmar, B.H.: Zwischen Sein und Sollen: Henry Mintzbergs Beitrag für die Managementwissenschaften. In: Organisationsentwicklung, vol. 4, issue 5, pp. 86 91 (2007)
23. Nadler, D., Tushman, M.: A Congruence Model for Organizational Assessment. In: Lawler III, E., Nadler, D., Cammann, C. (eds.): Organizational Assessment Perspectives on the Measurement of Organizational Behavior and the Quality of Work Life. John Wiley & Sons, pp. 261 348, New York et al. (1980)
24. Reiß, M.: Das Kongruenzprinzip der Organisation. In: Wirtschaftswissenschaftliches Studi um (WiSt), vol. 11, pp. 75 78 (1982)
25. Bekkers, W., van de Weerd, I., Brinkkemper, S., Mahieu, A.: The Influence of Situational Factors in Software Product Management: An Empirical Study. 2nd International Workshop on Software Product Management (IWSPM'08), pp. 41 48, Minneapolis/St. Paul (2008)
26. Clarke, P., O'Connor, R.: The situational factors that affect the software development pro cess: Towards a comprehensive reference framework. In: Journal of Information Software and Technology, vol. 54, issue 5, pp. 433 447 (2012)
27. Josten, F.: Determinanten von Product Management Strukturen Eine empirische Untersu chung in den USA; Europäische Hochschulschriften Band 227, Frankfurt/Main (1979)

Appendix: Literature on software product management tasks

Albourae, T., Ruhe, G., Moussavi, M. (2006), Lightweight Replanning of Software Product Releases, International Workshop on Software Product Management (IWSPM'06), Minneapolis/St. Paul, 2006, pp. 27 34
Barney, S., Aurum, A., Wohlin, C. (2008), A Product Management Challenge: Creating Software Product Value through Requirements Selection, in: Journal of Systems Architecture, 54, 2008, 6, pp. 576 593
Bebensee, T., van de Weerd, I., Brinkkemper, S. (2010), Binary priority list for prioritizing software requirements, 16th Requirements Engineering: Foundation for Software Quality (REFSQ'2010), Essen, 2010, pp. 67 78
Bekkers, W., Spruit, M. (2010), The Situational Assessment Method put to the test: Improvements based on case studies, Fourth International Workshop on Software Product Management (IWSPM'2010) Sydney, 2010, pp. 7 16
Bekkers, W., van de Weerd, I., Spruit, M., Brinkkemper, S. (2010), A Framework for Process Improvement in Software Product Management, in: Communications in Computer and Information Science Systems, Software and Services Process Improvement, 99, 2010, 1, pp. 1 12
Berander, P. (2007), Evolving prioritization for software product management, doctoral thesis at the School of Engineering, Blekinge Institute of Technology, Karlskrona 2007
Bjarnason, E., Wnuk, K., Regnell, B. (2010), Overscoping: Reasons and consequences A case study on decision making in software product management, Fourth International Workshop on Software Product Management (IWSPM'10), Sydney, 2010, pp. 30 39
Bjorner, D. (2011), Believable Software Management, in: Encyclopedia of Software Engineering, 1, 2011, 1, pp. 1 32
Cohen, G. (2010), Agile Excellence™ for Product Managers A Guide to Creating Winning Products with Agile Development Teams, Cupertino 2010
Dayani Fard, H. (2003), Quality based software release management, doctoral thesis at the Queen's University Kingston, Ontario 2003
Ebert, C. (2007), The impacts of software product management, in: Journal of Systems & Software, 80, 2007, 6, pp. 850 861
Fricker, S., Gorschek, T., Byman, C., Schmidle, A. (2010), Handshaking with Implementation Proposals: Negotiating Requirements Understanding, in: IEEE Software, 27, 2010, 2, pp. 72 80
Gorschek, T., Fricker, S., Palm, K., Kunsman, S. (2010), A Lightweight Innovation Process for Software Intensive Product Development, in: Software, IEEE, 27, 2010, 1, pp. 37 45
Gorschek, T., Gomes, A., Pettersson, A., Torkar, R. (2012), Introduction of a process maturity model for market driven product management and requ. eng., in: Journal of Software Maintenance and Evolution Research and Practice, 24, 2012, 1, pp. 83 113
Gorschek, T., Wohlin, C. (2006), Requirements Abstraction Model, in: Requirements Engineering Journal, 11, 2006, 1, pp. 79 101
Grynberg, A., Goldin, L. (2003), Product Management in Telecom Industry Using Requirements Management Process, IEEE International Conference on Software: Science, Technology and Engineering (SwSTE'03), Herzliya, 2003, pp. 63 70
Hanssen, G., Fægri, T. (2008), Process fusion: An industrial case study on agile software product line engineering, in: Journal of Systems & Software, 81, 2008, 6, pp. 843 854
Harness, D., Harness, T. (2004), The new customer relationship management tool product elimination, in: The Service Industries Journal, 24, 2004, 2, pp. 67 80
Helferich, A., Herzwurm, G. (2008), Softwaretechnische Ansätze für die Entwicklung flexibler Anwendungssysteme Ergebnisse einer explorativen Studie, Multikonferenz Wirtschaftsinformatik (MKWI'2008), München, 2008, pp. 1741 1752
Herzwurm, G., Pietsch, W. (2008), Guidelines for the Analysis of IT Business Models and Strategic Positioning of IT Products, Second International Workshop on Software Product Management (IWSPM'2008), Barcelona, 2008, pp. 1 8
Herzwurm, G.,Pietsch, W. (2009), Management von IT Produkten Geschäftsmodelle, Leitlinien u. Werkzeugkasten für softwareintensive Systeme u. DL, Heidelberg 2009

Humphrey, W. (2001), Winning with Software: An Executive Strategy, Boston u.a. 2001
IEEE Std 729 1983 (1983), IEEE Standard Glossary of Software Engineering Terminology, Inst. Electrical and Electronics Eng.
Iqbal, M., Zaidi, A., Murtaza, S. (2010), A New Requirement Prioritization Model for Market Driven Products Using Analytical Hierarchical Process, International Conference on Data Storage and Data Engineering (DSDE'10), Bangalore, 2010, pp. 142 149
ISPMA (2012), SPM Body of Knowledge, website: International Software Product Management Association, http://ispma.org/, last visit: 01.11.2012
Jagroep, E., van de Weerd, I., Brinkkemper, S., Dobbe, T. (2011), Implementing Software Product Portfolio Management, Fifth International Workshop on Software Product Management (IWSPM'2011), Trento, 2011, pp. 67 76
Jansen, S., Popp, K., Buxmann, P. (2011), The Sun also Sets: Ending the Life of a Software Product, Second International Conference (ICSOB'2011), Lecture Notes in Business Information Processing, Brussels, 2011, pp. 154 167
Khurum, M., Gorschek, T. (2011), A method for alignment evaluation of product strategies among stakeholders (MASS) in software intensive product development, in: Journal of Software Maintenance and Evolution Research and Practice, 23, 2011, 7, pp. 494 516
Kilpi, T. (1997a), New Challenges for Version Control and Configuration Management: a Framework and Evaluation, First Euromicro Conference on Software Maintenance and Reengineering (EUROMICRO'1997) Berlin, 1997a, pp. 33 41
Kilpi, T. (1997b), Product Management Requirements for SCM Discipline, in: Software Configuration Management, Lecture Notes in Computer Science, 1235, 1997b, pp. 186 200
Kilpi, T. (1998), Improving Software Product Management Process: Implementation of a Product Support System, Thirtyfirst Hawaii International Conference on System Sciences (HICSS'1998) Kohala Coast, 1998, pp. 3 12
Kittlaus, H. B., Clough, P. (2009), Software Product Management and Pricing, Berlin u. Heidelberg 2009
Konig, S. (2009), Finance as a Stakeholder in Product Management, Third International Workshop on Software Product Management (IWSPM'2009), Atlanta, 2009, pp. 15 22
Maedche, A., Botzenhardt, A., Neer, L. (2012), Software for People Fundamentals, Trends and Best Practices, Springer, Heidelberg u.a.
Mendonça, M., Bartolomei, T., Cowan, D. (2008), Decision making coordination in collaborative product configuration, Twentythird ACM Symposium on Applied Computing (SAC'08) Fortaleza, 2008, pp. 108 113
Mohamed, S., ElMaddah, I., Wahba, A. (2008a), Criteria Based Requirements Prioritization for Software Product Management, International Conference on Software Engineering Research & Practice (SERP'2008), Las Vegas, 2008a, pp. 587 593
Mohamed, S., ElMaddah, I., Wahba, A. (2008b), Towards Value Based requirements prioritization for software product management, in: International Journal of Software Engineering, 1, 2008b, 2, pp. 35 48
Mohamed, S., ElMaddah, I., Wahba, A. (2010), Criteria Based Framework for Software Product Management, in: International Journal of Software Engineering, 3, 2010, 1, pp. 29 52
Mohamed, S., Wahba, A. (2008), Value estimation for software product management, International Conference on Industrial Engineering and Engineering Management (IEEM'2008), Singapur, 2008, pp. 2196 2200
Müller, J. (2010), Preismanagementfür Software Produktlinien State of the Art, Website: Social Science Research Network (SSRN), URL: http://ssrn.com/abstract=1810973, last visit: 19.06.2012
Peine, K., Helferich, A., Schockert, S. (2012), Nachhaltige Anwendungssysteme dank IT Produkt management, in: Brandt Pook, H., Fleer, A., Spitta, T., Wattenberg, M. (eds.): GI Edition Nachhaltiges Softwaremanagement, LNes in Informatics, Bielefeld, 2012, pp. 36 49
Pfau, P. (1978), Applied Quality Assurance Methodology Proceedings of the Software, Quality Assurance Workshop on Functional and Performance Issues, California, 1978, pp. 1 8
Regnell, B., Kuchcinski, K. (2011), Exploring Software Product Management decision problems with constraint solving opportunities for prioritization and release planning, Fifth International Workshop on Software Product Management (IWSPM'2011), Trento, 2011, pp. 47 56
Saaksvuori, A., Immonen, A. (2008), Product Lifecycle Management, 3rd ed., Berlin and Heidelberg 2008
Sandusky, R., Gasser, L. (2005), Negotiation and the coordination of information and activity in distributed software problem management, International Conference on Supporting Group Work (GROUP'2005), Sanibel Island, 2005, pp. 187 196
Schackmann, H., Lichter, H. (2006), A Cost Based Approach to Software Product Line Management, First International Workshop on Software Product Management (IWSPM'06), Minneapolis, 2006, pp. 13 18
Shani, A., Sena, J. (2000), Knowledge management and new product development: learning from a software development firm, Third International Conference on Practical Aspects of Knowledge Management (PAKM'2000), Basel, 2000, pp. 19.1 19.5
Shinohara, Y., Dohi, T., Osaki, S. (1997), Comparisons of Optimal Release Policies for Software Systems, in: Computers & Industrial Engineering, 33, 1997, 3,pp. 813 816
Sneed, H., Hasitschka, M., Teichmann, M. (2005), Software Produktmanagement Wartung und Weiterentwicklung bestehender Anwendungssysteme, Heidelberg 2005
Striebeck, M. (2006), Ssh! We are adding a process…, Agile 2006 Conference (AGILE'06), Minneapolis, 2006, pp. 193 201
Suomalainen, T., Outi, O. S., Abrahamsson, P., Similä, J. (2011), Software product roadmapping in a volatile business environment, in: The Journal of Systems and Software, 84, 2011, 6, pp. 958 975
van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L. (2006a), On the Creation of a Reference Framework for SPM, First International Workshop on Software Product Management (IWSPM'2006), Minneapolis, 2006a, pp. 3 12
van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L. (2006b), A Reference Framework for Software Product Management, in: Technical report UU CS, 6, 2006b, 14, pp. 1 11
van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L. (2006c), Towards a Reference Framework for SPM, Fourteenth IEEE International Conference on Requirements Engineering (RE'2006), Minneapolis/St. Paul 2006c, pp. 319 322
van de Weerd, I., Katchow, R. (2009), On the integration of software product management with software defect management in distributed environments, Fifth Central and Eastern European Software Engineering Conference (CEE SECR'2009), Moskau, 2009, pp. 167 172
van Zyl, J. (2001), Process innovation imperative, Change Management and the New Industrial Revolution, First IEMC (IEMC '01), Albany, 2001, pp. 454 459
Zanker, M., Gordea, S. (2006), Measuring, monitoring and controlling software maintenance efforts, Thirteenth International Symposium on Temporal Representation and Reasoning (TIME'2006), Budapest, 2006, pp. 103 110

Towards Incorporating Sustainability while Taking Software Product Management Decisions

Birgit Penzenstadler[1], Mahvish Khurum[2], and Kai Petersen[2]

[1] Technische Universität München, Germany
penzenst@in.tum.de
[2] Blekinge Institute of Technology, Sweden
{Mahvish.Khurum|Kai.Petersen}@bth.se

Abstract. Software product managers are missing guidelines on how to incorporate different dimensions of sustainability in software product management and requirements selection decision-making. This is a challenge because considering sustainability perspective while selecting requirements has become a major objective for software product development companies; however, it is unclear how to support it during complex product management decision-making. In this paper, we identify the value aspects related to sustainability for software requirements selection. An exemplary dialogue between a consultant and a product manager illustrates how the proposed approach can be used while taking product management and requirements selection decisions. Our contribution provides software product managers with guidance on how to incorporate value aspects related to sustainability while taking software product management and requirements selection decisions.

Keywords: Sustainability, value-based software engineering, decision-making, software product management

1 Introduction

In today's world, software has become the main competitive advantage, enabling faster and cheaper innovation as well as product differentiation, and at the same time hardware is becoming standardized [10,35]. Simultaneously, the size and complexity of software in products are increasing, and so is the impact of software development decisions on the overall product offering [15]. That is, any decision taken regarding software, e.g. what features to realize, what quality to offer, or what technology to choose, will impact the entire product's life cycle and value, not to mention that it limits future possibilities and direction of the product and business (economic sustainability) [1,19]. Along with this, due to increased awareness about environmental, social and human sustainability, a challenging question is how a company can build innovative products that not only meet the needs of its customers, but are also built in a socially responsible and sustainable way? While it is required to build special software for the

customers to measure, monitor and act on various sustainability indices, it is equally important that the products are developed and managed in an adequate way with an in-depth understanding of the environment they are applied in. This situation gives rise to many decision-making challenges for industry practitioners, for example, which value aspects with respect to sustainability need to be considered while taking software management decisions? How does the realization of one functional or quality feature influence the sustainability value of the product offering, where short-term potential sales and revenues are almost always premiered over sustainability aspects? Answering these questions can help to innovate and develop products that do not only deliver value to the customers, but also enable development of products keeping in view the sustainability perspective. This perspective spans all levels of decision making: on the project, the product, and the portfolio level. Our research question is: *How can we incorporate sustainability as a primary objective with the conventional goals in software product management decisions?*

Value-based software engineering (VBSE) can help answering these questions as it emphasizes that every decision and/or feature of a product does not have an equal value like in a value-neutral setting [2]. This requires making decisions that are better for overall value creation, according to Kontio et al. [21] and Rönkkö et al. [29], and balancing short-term and long-term value creation.

Contribution The primary contribution of this paper is a list of value aspects that need to be considered from the perspective of sustainability while taking product management and development decisions. The Software Value Map is used as the basis for identification of these value aspects. The Software Value Map [20] provides a consolidated view on value aspects relevant for taking software product management and development decisions based on the Balanced Score Card approach. In addition to its application for sustainability concerns, a set of value aspects not yet covered by the Software Value Map has also been included, where each aspect is described and given a rationale. The identified value aspects can be used as criteria for taking requirements selection decisions. The application of the approach is illustrated in a fictitious dialogue between a product manager and a consultant.

2 Foundations and Related Work

The following sub-sections give a brief introduction to the concepts of sustainability and the Software Value Map as well as an overview of related work.

What is Sustainability? The four main dimensions of sustainability that we consider important are human, social, economic, and environmental, see Goodland [9]. The three latter ones are the dimensions known from the most cited definition of sustainable development by Brundtland et al. [4]: "... meets the needs of the present without compromising the ability of future generations to satisfy their own needs." The first dimension, human, is not present in the public discussion, but we argue that it should be included because it is the basis for the others.

Human sustainability: Human sustainability refers to the maintenance of the private good of individual human capital. The health, education, skills, knowledge, leadership and access to services constitute human capital. [9]

Social sustainability: Social sustainability means maintaining social capital and preserving the societal communities in their solidarity. Social capital is investments and services that create the basic framework for society. [9]

Economic sustainability: Economic capital should be maintained. The definition of income as the amount one can consume during a period and still be as well off at the end of the period can define economic sustainability, as it devolves on consuming value-added (interest), rather than capital. [9]

Environmental sustainability: Although environmental sustainability is needed by humans, it itself seeks to improve human welfare by protecting natural resources. These are water, land, air, minerals and ecosystem services; hence much is converted to manufactured or economic capital. Environment includes the sources of raw materials used for human needs, and ensuring that sink capacities recycling human wastes are not exceeded. [9]

Our analysis of how to incorporate sustainability into software product management decisions is based on these definitions as our understanding of sustainability. The foundation we use for guidance in taking software product management decisions is the Software Value Map, described in the following section.

The Software Value Map. The Software Value Map [20] provides a consolidated view of the software value concept utilizing four major perspectives: the financial, the customer, the internal business process, and the innovation and learning. The value aspects and value components contained in the map are collected through extensive review of economics, management and value-based software engineering literature.

The value map offers a unified view of value, which can be used by professionals to develop a common understanding of value, as well as acting as decision support to assure no value perspective is unintentionally overlooked when taking product management decisions. For example, during requirements selection in addition to short term increases in customer value and company revenue, a company's and product's long-term sustainability view can also be considered. While evaluating the effects of a requirement on the maintainability value of the product's architecture, human capital value of the company and innovation value would enable a comprehensive (long-term) impact analysis of a certain decision. Thus, by having a value focus, the overall trade-off between positive and negative impact on the present product offering can be estimated. This is central from many perspectives. For example, from a business perspective, the selection and realization of a feature might be good idea, but simultaneously the long-term effects pertaining to, e.g., sustainability of system architecture, might be very negative.

The taxonomy used to categorize the perspectives for measuring value was inspired by the balanced scorecard (BSC) approach, see Kaplan et al. [16,17]. BSC can be defined as a set of measures that gives managers a fast but comprehensive view of the business using four main perspectives, namely the financial,

customer, internal business process, and innovation and learning [16,17], each described below.

The *financial perspective* contains aspects that address the company's implementation and execution of its strategy which are contributing to the bottom-line improvement of the company. It represents the long-term strategic objectives of the organization and thus incorporates the tangible outcomes of the strategy in traditional financial terms [32,29]. Some of the most common financial measures that are incorporated in the financial perspective are earned value analysis and profit margins.

The *customer perspective* defines the value proposition that the company will apply to satisfy customers and thus generate more sales to the most desired (i.e. the most profitable) customer groups, see, e.g., Steven [32]. Measures that are selected for the customer perspective should measure both the value that is delivered to the customer (value proposition) with respect to the perceived value, which may involve time, quality, performance and service, and cost, and the outcomes that come as a result of this value proposition (e.g., customer satisfaction and market share).

The *internal process perspective* is concerned with the processes that create and deliver the customer value proposition. It focuses on all the activities and key processes required in order for the company to excel at providing the value expected by the customers both productively and efficiently. Quality, cycle time, productivity and cost are some aspects where performance value can be measured [16].

The *innovation and learning perspective* is the basis of any strategy and focuses on the intangible assets of an organization, mainly on the internal skills and capabilities. The innovation and learning perspective is the intellectual capital categorized as human capital, structural capital, and the organization capital of a company [16,23].

Related Work. The ISPMA provides a first body of knowledge, which does not explicitly consider sustainability yet [15]. Penzenstadler et al. [27] conducted a systematic literature review on sustainability in software engineering. The review revealed no work specifically related to sustainability in the context of software product management decision making. Cabot et al. [7] performed a case study for sustainability as goal for the ICSE organization with i* goal models to support decision making for future conference chairs, but don't discuss decision support for potential measures and do not provide methodical guidance or decision support. Naumann et al. [26] investigate how web pages can be developed with little environmental impact, i.e., energy-efficiently, but do not discuss implications on product management. Mahaux et al. [24] performed a case study on a business information system for an event management agency that advertises environment-friendly events but do not address the decision making challenge. While all of these works refer to sustainability goals, none of them discusses values related to sustainability and how to consider them while taking product management and requirements selection decisions. Moreover, within research on value-based software engineering, to the best of our knowledge, no

work yet explicitly discusses sustainability as one of the major consideration in software product management decision-making.

3 Identification of Value Aspects for Sustainability

The software value map can be used as a basis for identification of value aspects be considered from different sustainability perspectives, while taking product management and requirements selection decisions.
We discuss four sustainability dimensions and value aspects relevant to each of the dimensions are given with the rationale for their relevance to the discussed sustainability dimension along with the references for further readings. Tables 1-4 list the values (column *Value Name*) related to the sustainability dimensions, identify the balanced score card perspective they belong to (column *Perspective*), describe the value itself (column *Value Description*), why it matters for sustainability (column *Rationale*), and what can be done in order to improve it with references to further reading (column *Actions & Further Reading*). Please note that the list of identified value aspects is not complete rather it is the first attempt towards theoretical foundations for incorporating sustainability perspective while taking requirements selection decisions.

Human Sustainability. Software is developed and managed by people. Therefore, it is fundamental to consider value aspects related to human capital while taking product management decisions. The *human capital value* is described in Table 1. Although value aspects related to human sustainability are not necessarily related to all product management decisions, they actually influence the effects of other value aspects related to sustainability. For example, satisfied developers will be more focused to build high quality products with efficient use of resources.

Table 1. Value aspects for human sustainability

Value Name	Perspective	Value Description	Rationale	Actions & Further Reading
Human capital value	Innovation and learning	Human capital value refers to the stock of skills and knowledge embodied in the ability to perform labor so as to produce economic value. It is the value of skills and knowledge gained by a worker through experience	For human sustainability, human capital value should be increased by enhancing the skills and knowledge of the developers since they are the work force that develops the system.	Offering training and continued education improves skills and knowledge of an employee. Fitz-enz [8] provides measures for the economic value of employee performance.

Social Sustainability. For supporting social sustainability, a software development company may want to consider *customer capital value* while taking product management decisions. Furthermore, *network externalities* play a role in binding the customer. In addition to the values in Table 2 that are identified from the Software Value Map, there are other values that can be discussed in this context. The reason for not listing them is that they are related less directly to the software product under development, rather to the surrounding environment,

Table 2. Value aspects for social sustainability

Value Name	Perspective	Value Description	Rationale	Actions & Further Reading
Customer capital value	Internal Business perspective	Value of relationships that a firm builds with its customers, and which is reflected in their loyalty to the firm and/or its products. It is not reflected in a balance sheet in monetary terms.	Loyalty of customers is a stable basis for continuous bonding and customer retention [14].	Hennig-Thurau et al. [14] suggests improving bonding with customer with the use of motivation theory. Storbacka et al. [33] links service quality with customer satisfaction and profitability.
Network externalities	Customer perspective	The amount of other users of the software product that are relevant to the focal user, e.g., who might be motivated to use a service due to incentives for the user	If there are incentives for a user to motivate other people to use a service, the user might keep using it for two reasons: the incentive (e.g. lower costs), and the network of users that share the service [18].	Katz et al. [18] provide an analysis of the options to improve availability of complementary goods and services.

for example, values related to labor practices, human rights, society, and ethical behavior (see Silvius and Schipper [31]). Activities for good labor practices are to ensure employment, to work on labor-management relations, to provide training and education as well as organizational learning, and to offer diversity and equal opportunity. Human rights support is, e.g., to prevent discrimination. For society, potential measures are to improve community support, to perform adequate market communication, and to guard customer privacy [31]. Ethical behavior includes checking investment and procurement practices [31].

Economic Sustainability Within a software development context, value aspects related to economic sustainability need to be considered while taking product management decisions. The values detailed in Table 3 are *maintainability value*, *innovation value*, *differential value*, and *physical value w.r.t cost*. Economic sustainability is the aspect that is already most present in today's software business. One sub aspect of this is software sustainability, a term used interchangeably with software maintenance, and an innovation infrastructure are fundamental inputs to continuously maintain and evolve software products such that they sustain economically throughout their entire planned lifecycle. Moreover, competitive advantage has to be maintained to ensure economic sustainability of the company.

Environmental Sustainability Environmental sustainability may be improved by improving the *market requirements value*, the *physical value w.r.t. cost*, the *sustainability value of technology*, and the *product's intrinsic value*. Each of the values is detailed in Table 4. Selling software as services enables a higher rate of innovation and also reduces the number of expensive hardware upgrades that needs to be done. This, in turn, means an increased environmental sustainability. While it can go hand in hand with economic efficiency, this can also have cost-increasing effects, e.g., through additional activities.

Interrelationships Interrelationships in the value aspects identified from different sustainability dimensions are possible. The interrelationships can exist as the following effects:

Table 3. Value aspects for economic sustainability

Value Name	Perspective	Value Description	Rationale	Actions & Further Reading
Maintainability value	Internal business perspective	The capability of the software product to be modified. Modifications include improvements or adaptation of the software to changes in environment, and in requirements and specifications.	Maintainability of a software product is a foundation for sustainability [30] in a broader understanding, as evolution balances the factors to be accounted for when aiming at sustainability.	An approach for achieving software sustainability and how to measure it is given by Seacord [30].
Innovation value	Innovation and learning	The practical value of subject technology that is materialized in market (as a product or service) or in business process (as process innovation)	According to Hansen et al. [13], sustainability is a key driver of innovation. If they go hand in hand, innovation has to be supported.	Hansen et al.'s framework [13] allows also for conclusions with respect to the market.
Differential value	Internal business perspective	Differentiation is the process of distinguishing the differences of a product or offering from others, to make it more attractive to a particular target market. This involves differentiating it from competitors' products as well as one's own product offerings.	In order to have sustainable competitive advantage, it is fundamental to strive product features/capabilities that enable economies of development and/or lower profit margins (see Lado et al. [22]).	Hall [11] gives a framework for linking capabilities to sustainable competitive advantage.
Physical value w.r.t. cost (PVc)	Internal business perspective	A product being developed and marketed with lower development cost will have higher Physical value w.r.t. to cost. [20]	For economic sustainability, it is fundamental to keep the development costs as low as possible [6].	Byggeth [6] proposes a procedure for sustainability-driven design optimization illustrated with a case study.

- A positive impact on one value aspect might have positive impact on one or more sustainability dimension
- A negative impact on one value aspect might have negative impact on one more sustainability dimension
- A positive impact on one value aspect might have a negative impact on one or more sustainability dimension and vice versa

For example, if quality features, not even demanded by the customers, are provided; the intrinsic value of the product might be very high, however, this will negatively impact the environmental sustainability perspective as Physical value w.r.t. cost would decrease due to extra features produced (a waste). On the other hand, if generic products (which are demanded by majority of the customers) are developed and sold, Market requirements value would be high and Physical value w.r.t. cost would be high which doubles the positive impact on the environmental sustainability. Moreover, by developing maintainable products, while Maintainability value is increased which positively impacts the economic sustainability; however, this could have a negative impact on Human capital value because the developers feel that by maintaining the existing code no new skills are being learnt. Consequently, human sustainability is negatively impacted.
Resolving trade-offs between conflicting dimensions (often between economic and environmental, as environmentally sustainable involves adequate supplies) can only be solved by goal prioritisation—the economic side or the environment.

Table 4. Value aspects for environmental sustainability

Value Name	Perspective	Value Description	Rationale	Actions & Further Reading
Market requirements value	Internal business perspective	Represents the production value with respect to a given market requirement (Time & effort to implement a feature vs requirement's market demand & value)	Producing generic products can save resources (such as computers, electricity), as customized solutions may require additional environmental resources [12].	Murugesan [25] proposes principles and practices for green IT.
Physical value w.r.t. cost (PVc)	Internal business perspective	Represents the production value w.r.t. cost. A product being developed and marketed with lower development cost will have higher PVc	From environmental sustainability perspective, it should be ensured the resources are not wasted (adding to cost) during product development. However, efficiency can only contribute to, but not achieve sustainability by itself, see, e.g., Tomlinson et al. [34].	Silvius [31] takes transport, energy, waste, and materials into the resource balance. Poppendieck [28] proposes to eliminate "waste" in terms of partially done work, relearning, task switching, delays, defects etc.
Sustainability value of technology	Innovation and learning	The sustainability value of technology means how good or bad a technology is rated with respect to environmental impact due to its own production as well as usage during lifetime and later on for disposal.	If a new technology is being implemented, what is the environmental sustainability value of technology? Can it, e.g., increase interoperability possibilities to design more generic products/solutions?	Brown [3] provides insights and rationale to evaluate sustainability in technology for environmentally sound innovation.
Product's intrinsic value	Customer perspective	This includes functionality and quality attributes e.g. usability, reliability etc, of the product.	From environmental sustainability perspective, features and quality provided in the product has to be balanced w.r.t. resources used.	Byggeth [5] proposes a set of guiding questions for sustainable product development.

4 Illustrative Usage Scenario

The following dialogue is a fictitious discussion between Daniel and Mick. It illustrates the first steps of a usage scenario for a sustainability-driven application as depicted in Fig. 1. Mick is a product manager in a big car manufacturing company that wants to develop a car-sharing platform. Daniel is a method consultant from a well-established IT consulting company. They have already worked together in the past and generally get along well. The rich picture in Fig. 2 shows the most important elements of the car sharing platform. There is a community of users who can rent and share cars, there is a backend data base and there is a business infrastructure with maintenance, administration, and management. The speech bubbles indicate first starting points for the different aspects of sustainability. They first name the sustainability aspect and then, in parentheses, an exemplary respective value that can be considered for the car sharing platform.

Sustainability as Objective "*The vice-president tells me that we need to focus on sustainability for the development project of that new platform—so, how do I do that?*" Mick opens the discussion. "*Well, that depends on the goals you want to achieve with respect to sustainability.*" Daniel is a consultant well trained in first analyzing the problem and then developing a solution step by step with his customers, Fig. 1 Step 1. Mick sighs internally: "*That's not much*

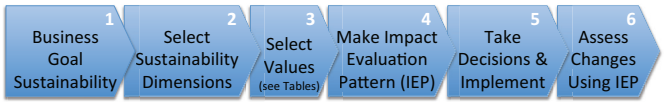

Fig. 1. Process Steps for Applying the SVM Sustainability Analysis

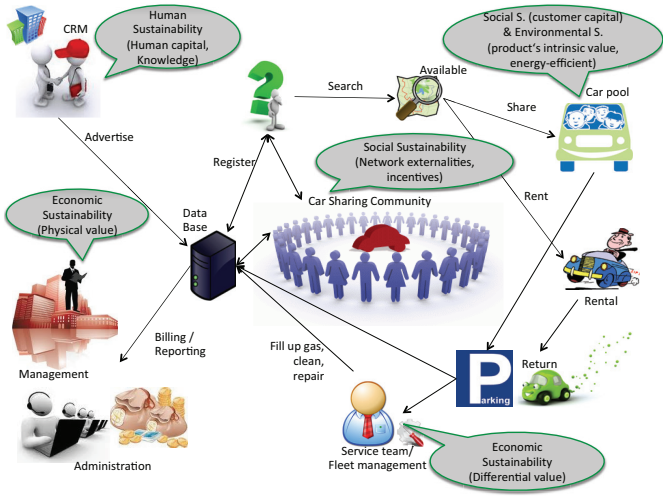

Fig. 2. Rich Picture of the Sustainability Aspects of the Car Sharing Platform

help—I don't know what goals I can set for sustainability. I don't even know of a precise definition what they refer to when using the term. To me, it seems like everybody has a different understanding of sustainability, so how can I come up with concrete goals for such a diffuse concept?"
Daniel is not surprised about this statement and provides a starting point: "*Okay, I do agree that many people might have a different understanding of what sustainability is, but luckily there are concrete definitions out there that we can use to make the concept more tangible in your context, for example the one given by Robert Goodland . . .*" and he quickly sketches the four dimensions of sustainability (given in Sec. 2): HUMAN, SOCIAL, ECONOMIC, and ENVIRONMENTAL, Fig. 1 Step 2.
"*I see . . .*" Mick acknowledges that it might be more than an abstract concept. Daniel takes that as an offer to further guide his customer: "*Sure, that is still quite abstract, but it is the most general goal you can start with for that particular dimension of sustainability. From here on, we can refine the goal like any abstract business goal your vice-president might come up with.*" Mick is still skeptical: "*I'm curious how you want to turn that into goals applicable to software-intensive systems development, but go ahead, we'll give it a try.*"
Daniel jumps up again to make more sketches on the whiteboard: "*We can look*

at that from the four different dimensions of sustainability." He starts questioning: "*What are current issues that need improvement? What are the values that are important here? What do your users want?*" Mick is a little surprised by that question: "*I guess they would want a good car sharing solution.*" "*That's a start. However, think about what values lie behind the need for a car sharing solution, for example, the wish to save the costs for a car and the wish to do something for the environment by saving energy?*" "*Good point*", says Mick, "*How can I structure my thoughts when trying to tackle the sustainability aspects for my product decisions and requirements elicitation?*"
"*I have that Software Value Map that gives a consolidated overview of common values for software-intensive product development. I have used it with a number of customers and it has proven sufficiently encompassing to be particularly helpful during analysis.*" And he hands him a one-page introduction to the SVM as provided in Sec. 2. Mick scans it quickly but then returns to the discussion at hand: "*Such a map is definitely useful for a start, but I need applicable guidance.*"

Human Sustainability "*Let's first take a look at the human sustainability dimension*", Daniel continues (Fig. 1, Step 3), "*According to the value map, the most important value that is relevant to be considered for human sustainability is* HUMAN CAPITAL VALUE. *Some of the metrics we can use to measure human capital are the general satisfaction of people and their impression of how their skills and knowledge develop over time. That would preserve human capital and therefore support human sustainability. So, how can we improve these two?*"
"*Okay, I see. Let me think*", Mick picks up the thought, "*User satisfaction depends on various factors, for example, the service costs and a good feeling when using the service. That's an issue for both our interface designers, who optimize the user interaction with the system and its services, and our economics guys, who calculate the service prices.*" "*And service level agreements like a high availability of cars et cetera.*" Daniel adds.
"*Sure, and if we want to support the improvement of their set of skills and knowledge, we could offer, for example, an education program at the time of registration. Could that be a start?*" Mick asks. "*Of course!*" Daniel replies, "*Most of your users will be aware of the basics but you can still provide them with more information on the specifics of your service and the impact on the environment.*"
"*Continuing that line of thought with knowledge and transparency,*" Mick extends the idea, "*we can perform an online evaluation of statistics and how much energy was saved in total, plus questionnaires that track even more, for example, which other means of transport they use apart from car sharing, and every user can optionally take part in that and gain knowledge on their individual statistics.*"
"*. . . and, thereby, offer to provide them with additional information, yes, good idea.*" Daniel agrees.
Mick starts taking notes on his To-Do template and scribbles facts here and there frowning at the piece of paper that is turning quite illegible. Daniel watches for a while and then proposes: "*We have developed a template for this purpose that we call Impact Evaluation Pattern—maybe you would like to make use of it?*"
"*Does that cost me extra?*" grumbles Mick. "*No*", shrugs Daniel, "*it's part of*

the service."
"*Simply put the concept of Impact evaluation patterns was inspired by software design patterns. In software engineering, a design pattern is a general reusable solution to a commonly occurring problem software design. The same philosophy can be used to identify Impact Evaluation Patterns in different decision-making scenarios. An Impact evaluation pattern can be described as a generally reusable solution for a commonly occurring decision-making challenge in a particular scenario. For example, a product manager can use an Impact Evaluation Pattern for initial screening from sustainability perspective to decide if a set of new requirements should be selected for implementation in the product or not.*"
On the second whiteboard in the room, Daniel sketches the template (Table 5, Fig. 1 Step 4): "*Here you can see the basic structure for documenting the impact evaluation pattern—which is part of what we are discussing right now. I'll keep explaining it while we continue.*"

Social Sustainability Mick settles for that for the moment: "*Okay, let's continue with the social dimension. What have you got on your value map for that?*" "*There are* CUSTOMER CAPITAL VALUES *and* NETWORK EXTERNALITIES. *How is the relationship that you build with your customers? Are they loyal?*"
"*They like our cars, and once they had one their likelihood to buy the same brand again is about 80%. I'm not sure though whether that applies to car sharing as well. We could establish a bonus system for frequent users.*" "*Good! How about* NETWORK EXTERNALITIES*?*" "*What is that supposed to be?*"
"*It means the amount of other users of the software product that are relevant to the focal user. Applied to a car sharing service platform, we have to think about incentives we can offer a user for spreading the word about our service and making other people use it.*" "*Phew, you mean like family discount and stuff? The problem is that we diminish our revenue, so the business analysts are always reluctant to give such bonuses. However, we will find something.*" "*Alright, I'll put it on the list.*"

Economic Sustainability "*Then let's talk about the economic dimension*", says Daniel, "*That's the one you might already have sorted out the most. The values are* MAINTAINABILITY, INNOVATION VALUE OF TECHNOLOGY *and* INNOVATION VALUE FOR MARKET, DIFFERENTIAL ADVANTAGE, *and* BUSINESS AGILITY." "*Yes, I guess we have elaborated on that for many hours.*"
"*Just what I thought, then why don't we directly move on to the environmental dimension? That was one selling point of your campaign draft, right?*" "*Yes, of course*", agrees Mick.

Environmental Sustainability "*Let's see what your value list says*", Mick continues, "MARKET REQUIREMENTS VALUE, PHYSICAL VALUE WITH W.R.T. COST, *and so on. These first two are about resource saving, right? That would increase the respective values.*"
"*Yes*", Daniel agrees, "*one of your market requirements would be to have an environmentally sustainable service, and you can consider various aspects for that—transport, i.e., local procurement, digital communication, traveling, and transport; then energy, i.e., energy used, and emission / CO2 from energy used;*

waste, i.e., recycling and disposal; and materials, i.e., reusability, incorporated energy, and resources. The other part of resources and potential 'waste' to be considered are the working hours—on one hand spent in design and development and on the other hand as 'waste' in terms of partially done work, extra features, relearning, handoffs, task switching, delays, defects, etc."
"*That list is even longer than the one we calculated from—I will double check that with our business analysts. For a start, the transport in our service refers mainly to the vehicle availability. If there aren't enough vehicles in a hot spot area, the service personnel have to move the vehicles accordingly. We definitely want to avoid that because it is costly in terms of emission and money. What is it with the* SUSTAINABILITY VALUE OF TECHNOLOGY*?*" Mick inquires.
Daniel replies: "*You are potentially decreasing emissions by decreasing traffic—and I'm sure you will be able to find a lot more within an analysis of environmental optimization potential. You also have to evaluate how the desired system quality might affect the environment in a negative way, for example, by putting a lot of cars out there to ensure availability, you make a considerable impact again on the environment.*" "*True, we'll have to perform an analysis on how they contradict.*"

Table 5. Impact evaluation pattern for sustainability aspects

Pattern Name	Impact eval. on sustainability aspects pattern for a product manager
Intent and Motivation	To perform a detailed value analysis with respect to sustainability.
Applicability	Analyze features to include in a product w.r.t. sustainability
Value aspects	Value aspects listed in Table 1-4
Impact evaluation criteria	see `http://www.bth.se/tek/aps/mkm.nsf/pages/softaware-value-map`
Consequences	//to be added when the pattern is put into actual use
Involved stakeholders	Product manager, domain expert, sustainability expert, process engineer, project manager

Impact Evaluation Pattern "*Now we have a lot of scribbled notes on the whiteboard—what you called Impact Evaluation Pattern earlier on.*" Mick gets back to the sketch on the whiteboard. "*Yes.*" Daniel emphasizes, "*With the help of such impact evaluation patterns, the company can have tremendous benefits. An impact evaluation pattern presents a consolidated view of value aspects to be considered while deciding with respect to sustainability. Furthermore, it provides a common understanding and vocabulary, as well as acting as decision support to assure no value aspect is unintentionally overlooked. And finally, it enables a conscious impact evaluation (positive and/or negative) of relevant value aspects.*" Mick likes decision support: "*That's good. And I can teach my staff to use the guideline instead of trying to mentally infuse my experience into them, as genuine experience cannot really be passed on.*" "*True*", Daniel adds, "*and, furthermore, we can define corresponding rubrics that help to evaluate that selected measures, so you and your VP can see the direct impact.*"
"*Rubrics? What would be an example for that?*" "*For example, for* HUMAN CAPITAL VALUE*, you can assess general satisfaction, time per year spent on continued education, and employee fluctuation.*" "*Okay, that makes it assessable for management—for these metrics, we already have some kind of reporting, so I know where to get the data from.*" Daniel wraps up the discussion. "*Alright, I*

hope I could show you how you can use the value map to identify optimization potential with respect to the different dimensions of sustainability." "*Yes, thanks, I feel quite prepared now for the next meeting with the vice president.*"

In the meeting with the vice president, Mick will realize Step 5 of the process in Fig. 1: the vice president will take the decisions and Mick is responsible for their implementation. The Impact Evaluation Patterns will be reused for the assessment of Step 6.

5 Discussion: Transfer to Practice

The presented usage scenario is the first attempt to illustrate the approach and can by no means substitute the evaluation in a sufficiently sized industrial case study. The preparation of such a case study is under way, but as we are looking for academic feedback in parallel we offer our concepts for early discussion.
We believe the approach is promising since its is based on a theoretically solid and empirically evaluated Software Value Map. The Software Value Map and impact evaluation pattern have been used in a case study at Ericsson for identifying value aspects to be considered for requirements selection from different stakeholders' and they have been proven usable and useful. The industry practitioners did not only verify the benefits of having a consolidated view of value components, relevant for a particular Impact evaluation pattern, for decision-making; they also found the Software Value Map a step towards common definitions and understanding of value components enabling effective communication [20].
We expect to implement a similar case study for a sustainability analysis with equally positive results.
One important issue for consideration are conflicts that arise between different dimensions, as already mentioned in "Interrelationships" on p. 6. The explicit catalogue of values provides means to identifying such conflicts. However, the question of how such trade-offs can be solved while planning requires the prioritisation of goals. Which dimension will be considered most important in our future requires solutions in much broader terms than the approach at hand.

6 Summary and Future Work

In this paper, we have presented an approach to incorporate sustainability related value aspects while taking software product management decisions on the project, product, or portfolio level. Its usage is illustrated in a scenario where there is a dialogue between a consultant and a software product manager.
As a first effort to factor sustainability as primary aspect in value-based software engineering, the approach still needs to evolve and be explored. The proposed list of sustainability related value aspects is ready to use but not necessarily comprehensive; value aspects and perspectives can be complemented as needed. Our next step is to evaluate the approach in an industrial setting with adequate complexity to gain resilient feedback on its application in practice.
Future work is to improve upon how the identified sustainability aspects should

be measured in industry.

Acknowledgement This work is part of the EnviroSiSE project (grant: PE2044/1-1) funded by the DFG in Germany and the BESQ+ research project funded by the Knowledge Foundation (grant: 20100311) in Sweden.

References

1. A. Aurum, C. Wohlin, and A. Porter. Aligning software project decisions: A case study. *International Journal of Software Engineering and Knowledge Engineering*, 16:795–818, 2006. 6.
2. S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Grunbacher. Value-Based software engineering. Springer, Germany, 2006.
3. M. T. Brown and S. Ulgiati. Emergy-based indices and ratios to evaluate sustainability: monitoring economies and technology toward environmentally sound innovation. *Ecological Engineering*, 9(1-2):51–69, 1997.
4. Brundtland et al. Our common future, 1987. United Nations Conference on Environment and Development (UNCED).
5. S. Byggeth, G. Broman, and K. Robert. A method for sustainable product development based on a modular system of guiding questions. *Journal of Cleaner Production*, 15(1):1–11, 2007.
6. S. H. Byggeth, H. Ny, J. Wall, G. Broman, and K.-h. Robèrt. Introductory procedure for sustainability-driven design optimization. 2008.
7. J. Cabot, S. Easterbrook, J. Horkoff, L. Lessard, S. Liaskos, and J.-N. Mazon. Integrating sustainability in decision-making processes: A modelling strategy. In *Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, pages 207 –210, May 2009.
8. J. Fitz-enz. *The ROI of human capital: Measuring the Economic Value of Employee Performance*. AMACOM Division American Management Association, 2000.
9. R. Goodland. Sustainability: Human, social, economic and environmental. In *Encyclopedia of Global Environmental Change*. John Wiley and Sons, 2002.
10. T. Gorschek, S. Fricker, K. Palm, and S. Kunsman. Star search - a light-weight innovation process for software intensive product development. *IEEE Software*, 27:37–45, 2009. 1.
11. R. Hall. A framework linking intangible resources and capabiliites to sustainable competitive advantage. *Strategic Management Journal*, 14(8):607–618, 1993.
12. S. Hallstedt, H. Ny, K. Robèrt, and G. Broman. An approach to assessing sustainability integration in strategic decision systems for product development. *Journal of Cleaner Production*, 18(8):703–712, 2010.
13. E. G. Hansen, F. Grosse-Dunker, and R. Reichwald. Sustainability innovation cube - a framework to evaluate Sustainability-Oriented innovations. *SSRN eLibrary*, 2009.
14. T. Hennig-Thurau and U. Hansen. *Relationship marketing: gaining competitive advantage through customer satisfaction and customer retention*. Springer, 2000.
15. International Software Product Management Association. Product Management in the Business of Software, BoK 1.1, 2013. `http://ispma.org/spmbok/`.
16. A. S. Kaplan and D. P. Norton. *The balanced scorecard: translating strategy into action*. Harvard. Business School Press, Boston, United States of America, 1996.
17. R. S. Kaplan and D. P. Norton. The balanced Scorecard—Measures that drive performance. *Harvard Business Review*, 1992.

18. M. L. Katz and C. Shapiro. Network externalities, competition, and compatibility. *The American Economic Review*, 75(3):424–440, June 1985.
19. M. Khurum, K. Aslam, and T. Gorschek. MERTS — a method for early requirements triage and selection utilizing product strategies. pages 97 – 104, Nagoya Japan, 2007. IEEE Computer Society.
20. M. Khurum and T. Gorschek. Software value map - an exhaustive collection of value aspects for the development of software intensive products, 2010. Khurum2011.
21. J. Kontio, J. Warsta, M. M. Makela, M. Ahokas, P. Tyrvainen, and P. Poyry. Software business education for software engineers: Towards an integrated curriculum. Institute of Electrical and Electronics Engineers Inc., 2006.
22. A. A. Lado, N. G. Boyd, and P. Wright. A Competency-Based model of sustainable competitive advantage: Toward a conceptual integration. *Journal of Management*, 18(1):77 –91, Mar. 1992.
23. J. Liebowitz and C. Y. Suen. Developing knowledge management metrics for measuring intellectual capital. *Journal of Intellectual Capital*, 1:54–67, 2000. 1.
24. M. Mahaux, P. Heymans, and G. Saval. Discovering Sustainability Requirements: an Experience Report. In *17th International Working Conference on Requirements Engineering: Foundation for Software Quality*, 2011.
25. S. Murugesan. Harnessing green IT: principles and practices. *IT Professional*, 10(1):24–33, Feb. 2008.
26. S. Naumann, M. Dick, E. Kern, and T. Johann. The greensoft model: A reference model for green and sustainable software and its engineering. *Sustainable Computing: Informatics and Systems*, (0):–, 2011.
27. B. Penzenstadler, V. Bauer, C. Calero, and X. Franch. Sustainability in software engineering: A systematic literature review. In *International Conference on Evaluation and Assessment in Software Engineering (EASE)*, 2012.
28. M. Poppendieck and T. Poppendieck. *Implementing Lean Software Development - from concept to cash*. Addison-Wesley, 2008.
29. M. Rönkkö, C. Frühwirth, and S. Biffl. Integrating value and utility concepts into a value decomposition model for Value-Based software engineering. In *Product-Focused Software Process Improvement*, pages 362–374. Springer, 2009.
30. R. Seacord, J. Elm, W. Goethert, G. Lewis, D. Plakosh, J. Robert, L. Wrage, and M. Lindvall. Measuring software sustainability. 2003.
31. G. Silvius and R. Schipper. A maturity model for integrating sustainability in projects and project management. Istanbul, 2010.
32. B. D. Steven. Using the balanced scorecard process to compute the value of software applications. In *ICSE'06 Proceedings of the 28th international conference on software engineering*, Shanghai, China, 2006. ACM.
33. K. Storbacka, T. Strandvik, and C. Grönroos. Managing customer relationships for profit: The dynamics of relationship quality. *International Journal of Service Industry Management*, 5(5):21–38, Dec. 1994.
34. B. Tomlinson, M. S. Silberman, and J. White. Can more efficient IT be worse for the environment? *IEEE Computer, Green IT column*, 44(1), Jan. 2011. TSW2011.
35. K. Wnuk, R. B. Svensson, and B. Regnell. Investigating upstream versus downstream decision-making in software product management. In *Proceedings of the 2009 Third International Workshop on Software Product Management*, IWSPM '09, pages 23–26, Washington, DC, USA, 2009. IEEE Computer Society.

Mobile Game Software Product Management

David Callele

Department of Computer Science, University of Saskatchewan
Saskatoon, Saskatchewan, Canada
callele@cs.usask.ca

Abstract. This work describes a case study wherein observations of a greenfield mobile game software product line development effort were gathered to help identify directions for future work. Business and technology requirements were gathered by the development team and the consequences of adopting third-party middleware as the basis for a software product line were explored and the management of development efforts and product differentiation were identified as areas for future work. The effects of adopting commercially available middleware upon the requirements for mobile game SPLs are discussed and the need for requirements engineering to focus upon the customer experience in this context is identified.
Keywords: Software product line, requirements, mobile games, software product management

1 Introduction

Commercial software developer interest in the smartphone application market is significant. The availability of application stores (*a.k.a.* app stores) for product distribution has bypassed much of the traditional retail channel (with the associated negative effects upon the viability of the retail channel). This alternative market access has removed many barriers to new product entry and has quickly been exploited by mobile game developers – it is widely reported that games are the most commonly downloaded smartphone apps. Some games require the payment of a fee before the customer can download the game. Other games adopt a "try before you buy" format; the customer can start playing for free but after an initial demo they have to buy the game to continue to play. To generate revenue, the game producer must convert the customer during the critical free-play period. Some games adopt the "freemium" model where continued gameplay is free but is greatly facilitated (*e.g.* progress is accelerated) if the customer makes in-game purchases of gameplay related items.

The transactional price of a typical mobile game is now ingrained at $0.99, some fraction of the price of a single cup of coffee in North America – in the same way as a cup of coffee is a discretionary purchase of a mass-market consumable item, so is the purchase of a game. While some games have reportedly generated more than $1M in revenue, the average revenue is closer to approximately $1,000 per app (as reported by Vision Mobile, *Mobile Developer Economics*, available at http://developereconomics.com) – far less than the amount necessary for commercial viability. Mobile game development is, therefore, a significant financial gamble and attempts to mitigate the financial

risk posed by low revenues are likely to put pressure upon development budgets and development timelines. Software product management [5] will be under pressure to minimize investments in process, potentially promoting process evolution in an ever more lightweight direction which could place significant challenges upon the process of defining and scoping mobile games. One way to mitigate this risk could be to adopt an SPL approach to development efforts.

We report herein upon the results of a case study directed toward gathering and reporting empirical observations of mobile game software development (with a modest emphasis on requirements issues) to more clearly identify directions for future work. The author, a RE practitioner and software developer, was embedded within the development team and the reported observations have an action research bias.

2 Related Work

Investigations into the role of requirements in general videogame development [3] have noted that the traditional output of the preproduction process in game development (the Game Design Document (GDD)) was often misused as a (weak) requirements specification. Furtado *et al.* [7] apply Domain Analysis to digital game Software Product Line (SPL) development, advocating the creation of a "game design vision" that guides the Domain Analysis. They explicitly identify the elements that will, and will not, be part of the SPL, drawing these elements from the Game Design Document or its equivalent – noting that traditional Requirements Engineering cannot be applied *as is* to game development [7]. Finally, Furtado *et al.* [7] recognize, as we did [4], that software product management must consider the experience aspects of game requirements.

In complementary work, Blow noted that game development [2], requires an unusually large breadth of engineering knowledge to define functional requirements. However, the technical difficulties stressed by Blow in 2004 were later addressed by experience gained in the development of very-large scale games. On a smaller scale, mobile game development requires meeting the challenges identified by Blow *plus* meeting the market constraints identified in the introduction to this work.

Folmer [6] investigated and reported on the potential for component based game development (*e.g.* rendering engines) as a means of addressing rising development costs. Folmer derived a reference architecture for the game domain, but this work was left behind by market forces with the advent of multi-platform game engine middleware (although they do make note of game engines in their concluding statements, stating that they could be potential "domain frameworks"). Finally, agile development methodologies are often suggested when faced with tight time-to-market constraints. Kanode and Haddad [8] advocate the use of agile methods during preproduction to help ensure that the resulting gameplay experience is as intended. However, they report that a thorough preproduction process (the game development analog to a more-than-cursory requirements process) appears to negate the expected benefits from the use of agile development methodologies.

In *The Elements of Software Product Management* [1] we note that Chapter 3 (Software as a Business) does not address games as a unique category, only touching upon freeware as an alternative business model. Chapter 4 (The Elements of Software Prod-

uct Management) describes computer games as low-priced products that enjoy a burst of popularity then fade away reducing the need for longer-term planning, development is further characterized as narrative-driven and following a movie production paradigm rather than a feature-oriented development process. Chapter 5 (The Elements of Software Pricing) investigates only console games (typically priced around $50 or more), considering them consumer commodity products that are not priced by their value but rather by customer willingness to pay. Mobile games at the low price points mentioned herein are not considered in this work.

3 Case Scenario and Observations

This work is based upon practical experience gained and observations made [9] during a mobile game Software Product Management development effort. The development team had five members in addition to the author, three of which had significant game production and development experience gained by participation in more than 10 commercial game developments, typically with budgets in the two to five person year range. The long-term business goal for the SPM development effort studied herein targets the following relative expenditures for elements of the game development effort: (1) increase the budget for gameplay design from 10% to 20%, (2) reduce the software development budget from approximately 50% to approximately 10% by utilizing SPL techniques for the Core, (3) increase the maximum budget for scripting of media assets from 10% to 40%, and (4) increase the budget for media asset development from 50% to 70%. The team began by identifying the set of business and technology requirements for their effort. Note that all of these goals are stated in financial terms in recognition of the market risks identified earlier.

The development team modeled the business requirements and stakeholders as shown in Figure 1. The different stakeholder groups have distinct goals and widely divergent planning timelines. In the short-term, the focus is upon the delivery of a particular game instance while in the long-term, the focus is on maintaining and enhancing the SPL assets for the organization: both the software assets (the Core) and the media assets. The role of the SPL product manager follows the model established by the role of a producer in the movie industry, a market-aware role that spans all three stakeholder groups with executive authority over all aspects of the development effort. We note that the end customer is buying an entertainment experience, not a software product, and the software exists only to enable the entertainment experience.

Preproduction management is focused upon the requirements for the specific instance of the game currently under development. They are concerned with defining the game experience (without concern for implementation details) and tend to concentrate upon requirements for the media assets and how they contribute to the look and feel of the game.

Production management is concerned with the current game (the short-term tactics of delivery) *and* with the contribution the game can make to the SPL asset base (maintaining the long-term strategy of media and Core contributions to the SPL assets). Their focus on the need to deliver means that these stakeholders are most likely to address the

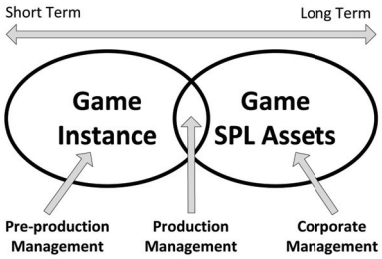

Fig. 1. Stakeholder requirements focus

traditional domain of requirements engineering: the specification of the functional and non-functional requirements for the software development effort.

Corporate management is focused upon the business requirements. In the specific instance of a game development project they are concerned with financing the current project (development and marketing) and maintaining budget control. This implies a set of market requirements (such as game genre and style guidelines) that are not often reported upon in the requirements engineering literature; requirements such as understanding the market for the game are critical precursors to preproduction. Given the market constraints upon these enterprises, understanding which requirements are mandatory and maintaining focus upon addressing *only* those requirements is an important contributor to long-term success given the low probability of success in this market.

In the general instance of the game SPL assets, these assets reflect the need to be commercially viable. The higher the proportion of the asset base that has been used (rather than *built but not used* in the release version of the game) or reused, the better the return on investment. Further, these assets may also represent a significant portion of the value of the business entity.

The aforementioned market pressures have lead sector participants to investigate any means available to get to market as inexpensively and as quickly as possible. The team considered restricting the Core to the target market (*e.g.* iOS or Android) and building upon the native libraries. They also considered implementing a completely proprietary Core for each of the target markets, perhaps building upon third-party libraries. Faced with the noted market pressures, they finally committed to a third-party multi-platform middleware game engine as the basis for the Core (http://www.unity3d.com).

4 Observations

The alignment between the expected implementation of the requirements for the greenfield SPL were sufficiently close to that of a particular third-party middleware game engine that the decision was made that it was “close enough”. While the middleware does reduce the time to market, it also induces a dependency upon the middleware provider that may be a substantial business risk. Careful management is indicated.

As the capabilities of middleware offerings continue to grow, offering more features (depth) and more supported platforms (breadth), they have become the basis for many mobile game SPLs. They represent the accumulation of significant domain experience and evolve just as the core of any SPL is expected to evolve. Their abstractions define the architecture for any game implemented upon them and their APIs represent the core functional requirements for the software artifact.

Development efforts based upon such middleware must depend more heavily upon the playing experience than technological capabilities to achieve market differentiation. The middleware can also impose constraints upon game designs: If a game concept requires a feature not available in the middleware, it may be cut due to budget and/or time constraints.

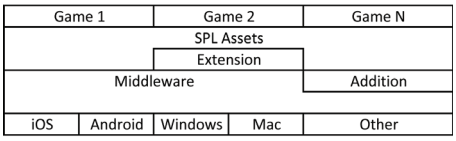

Fig. 2. Architectural Model

Once the middleware platform was adopted, it was our experience that game SPL assets evolved, as shown in Figure 2, to become: (1) extensions to the middleware (build upon the middleware to increase the *depth* of possibilities), (2) additions to the middleware (build beside the middleware to increase the *breadth* of possibilities), and (3) proprietary SPL Assets including reusable software and media assets of all types.

5 Conclusions and Future Work

Effective software product management is a key success factor in mobile game development. We have observed the process of introducing SPL planning into a game development effort and note the following for further consideration in future work.

- The adoption of a game engine as the core of the SPL creates a need for a software product management model for middleware based mobile game software product lines. While there are analogs in the adoption of enterprise platforms like SAP, the addition of media assets to the production effort creates unique challenges.
- The ROI for adopting middleware as the core of a SPL should be extensively evaluated as there may exist special considerations due to the relative size of investments for middleware *vs.* extensions, additions, and other SPL assets.
- Management decision support (beyond ROI) is needed for evaluating the effect of extensions *vs.* additions to middleware based SPL when meeting market pressures.
- Mechanisms to identify and control product differentiation in middleware based SPLs are needed as the scope for technology-based product differentiation is reduced by this dependency.

A risk management approach that is aware of the market characteristics and constraints discussed herein would likely lead to directing requirements engineering and software product management efforts in mobile game development to focus upon "what is possible within the budget" rather than on "things that we would like to have" – especially in the absence of evidence that the inclusion of a new feature or media asset will lead to greater probability of market success. There is little margin for error and there is no guarantee that, even if the requirements are correct, the game will be a market success. Moreover, adopting third-party middleware and executing on third-party hardware gives a scant opportunity for technology-based product differentiation. New software features will remain exclusive for only a single product cycle – competing on this basis is very difficult and places developers in a constant state of technological warfare with their competition.

If the requirements engineering effort is only about determining the functional and non-functional requirements for the software artifact then it appears that our contributions to this domain are destined to shrink, perhaps even to the point of near irrelevancy as software development consumes an ever smaller portion of the development budget. However, if we understand that the customer experience is the differentiator, and we adapt and extend requirements engineering to support this *modus operandi* with techniques such as experience requirements [4], our ability to contribute remains strong.

References

1. *Software Product Management and Pricing*. Springer Berlin Heidelberg, 2009.
2. Jonathan Blow. Game development: Harder than you think. *Queue*, 1:28–37, February 2004.
3. David Callele, Eric Neufeld, and Kevin Schneider. Requirements engineering and the creative process in the video game industry. In *RE '05: Proceedings of the 2005 13th IEEE International Requirements Engineering Conference*, pages 240–250, Paris, France, 2005. IEEE Computer Society.
4. David Callele, Eric Neufeld, and Kevin Schneider. Introducing experience requirements. In *RE '10: Proceedings of the 2010 18th IEEE International Requirements Engineering Conference*, Sydney, Australia, 2010. IEEE Computer Society.
5. Christof Ebert. The impacts of software product management. *J. Syst. Softw.*, 80:850–861, June 2007.
6. Eelke Folmer. Component based game development: A solution to escalating costs and expanding deadlines? In Heinz Schmidt, Ivica Crnkovic, George Heineman, and Judith Stafford, editors, *Component-Based Software Engineering*, volume 4608 of *Lecture Notes in Computer Science*, pages 66–73. Springer Berlin / Heidelberg, 2007.
7. Andre Furtado, Andre Santos, and Geber Ramalho. Streamlining domain analysis for digital games product lines. In Jan Bosch and Jaejoon Lee, editors, *Software Product Lines: Going Beyond*, volume 6287 of *Lecture Notes in Computer Science*, pages 316–330. Springer Berlin / Heidelberg, 2010.
8. C.M. Kanode and H.M. Haddad. Software engineering challenges in game development. In *Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on*, pages 260 –265, april 2009.
9. R.K. Yin. *Case Study Research: Design and Methods*. Sage Publications, 2003.

Assessing the Strategy Relevance of Delta Requirements
A Bottom-up Feature Generation Approach

Gabriele Zorn-Pauli , Barbara Paech

University of Heidelberg, Institute of Computer Science, Im Neuenheimer Feld 326, 69120 Heidelberg, Germany
{zorn-pauli, paech}@informatik.uni-heidelberg.de

Abstract. Strategic release planning is applied to decide which features to implement in which release to ensure competitive advantage. Therefore, continuously arriving changes to requirements at different levels of abstraction have to be aligned with business strategies. Especially, incrementally developed software systems have to cope with a flow of incoming delta requirements that specify enhancements to existing functionality. It is challenging for product managers to relate delta requirements to business strategies and to assess whether a delta requirement is relevant to business strategies scheduled for a specific release. The major idea of the proposed bottom-up feature generation approach is to handle these delta requirements at feature level to relate them more effectively to business strategies. The contribution of this paper is two-fold. First, a more detailed description of the already introduced Requirement Abstraction and Solution Model (RASM) to address requirements and solution abstraction. Second, a bottom-up feature generation approach that maintains RASM and which helps to assess the business strategy relevance of continuously arriving delta requirements.

Keywords: strategic release planning, roadmapping, feature generation, requirement bundling, delta requirement classification, requirement abstraction

1 Introduction

The development of software systems is characterized by continuous change to requirements at different levels of abstraction, 'responding to evolving requirements, platforms, and other environmental pressures' [2]. Passos et al. [7] postulated a vision that feature orientation of software design and of the software development process is able to handle change more effectively than previous methods. They stated the controlling and executing of change as the major challenge for most software projects. To align these changes with business strategies, strategic release planning (SRP), also called roadmapping, is applied to decide how features are scheduled for different releases to ensure competitive advantage. Therefore, SRP translates business strategies top-down into *business features* (BF), linking the business view with requirements engineering [6].

In an industrial case study [8], we explored that SRP in practice generates features also bottom-up by bundling continuously arriving delta requirements into *software features* (SF). These software features represent 'cohesive bundles of requirements ad-

dressing important capabilities of the system' [1]. *Delta requirements* specify enhancements to existing system functionality at a low level and require a relation to the existing system to provide an understanding of the delta [4]. Without knowing the relation between delta requirements and the existing system, it is difficult for release planning teams or product managers to understand the impact of delta requirements on existing BF structures and further release plans. Additionally, for a software product that is used globally, customers from different countries raise delta requirements that are motivated by their country-specific business strategies and processes. Another challenging task is to recognize delta requirements duplicates in large-scale projects.

This paper provides two contributions. First, a more detailed description of the already introduced Requirement Abstraction and Solution Model (RASM) [8] to handle requirements and solution abstraction during SRP. Second, a bottom-up feature generation approach that maintains RASM and which helps to assess the business strategy relevance of continuously arriving delta requirements.

The remainder of this paper is structured as follows: Section 2 provides a detailed description of the RASM using an application example. In Section 3, the bottom-up feature generation approach steps are introduced. Section 4 provides a discussion of RASM application and open issues. Finally, Section 5 concludes the paper and provides an outlook to future work.

2 RASM

In [8], we reported on an industrial case study to analyze the SRP process of a company in the health care domain, whereby we gathered four major requirements for feature generation support: *(FG1) support top-down and bottom-up feature generation, (FG2) support the aggregation of relevant changes into existing release plans, (FG3) support delta requirements handling, and (FG4) support feature classification and variability.*

The RASM has already been introduced briefly in [8] as a preliminary solution proposal to address these requirements. It extends the Requirement Abstraction Model (RAM) [3] by distinguishing explicitly between BF and SF to support top-down and bottom-up requirement and solution abstraction. In this section, we describe a RASM application example using real data from the company.

Business Strategies such as *Deal Compliance & Monitoring* and *Pricing Tool Enhancements*, shown in Figure 1 (a), represent business case initiatives with the different priorities 9 and 7, in a range from 1 (extremely low) to 9 (extremely high). In most cases, companies pursue several strategies at the same time and SRP aims at selecting features that optimally support highly ranked business strategies. Therefore, each priority change has an impact on existing release plans, which makes it necessary to keep business strategy priorities up-to-date.

Business Features (BF) represent business strategies at product level. The example provided in Figure 1 (a) illustrates two BFs derived in a top-down manner: *Instrument Freight Handling* and *Bloodgas Business*. They are related to a business strategy and describe business requirements at a high-level independently of the existing software system. In a globally operating company, there are several company sites in different countries (Ci) that have varying priorities (Pi) for BFs based on, for example different markets. To successfully apply SRP for a globally used software system, these BFs

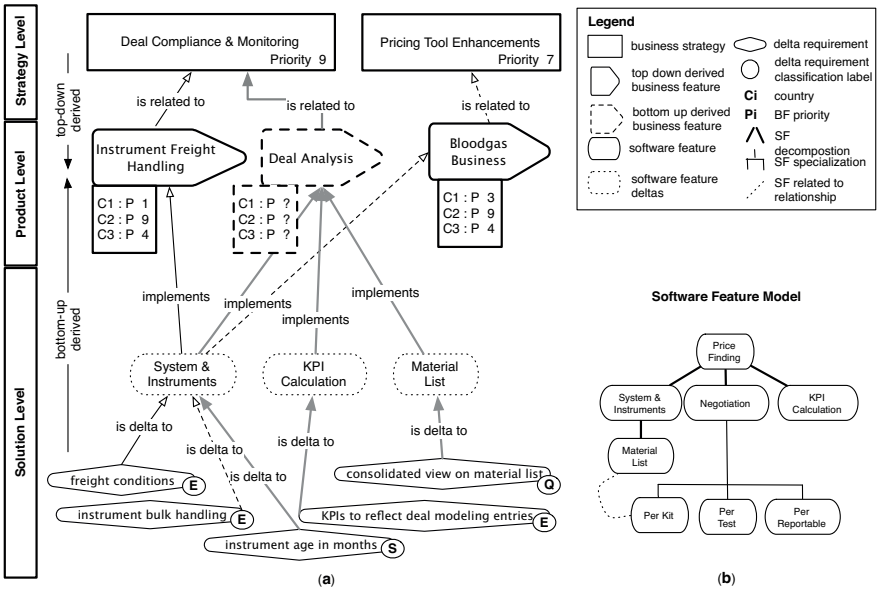

Fig. 1. (a) RASM application and (b) SF model example

have to be implemeted and provided first that guarantee the largest common overlap with respect to highly prioritized BFs by the different countries. Therefore, the country priorities of BFs have to be kept up-to-date and BFs must have a relation to one or more business strategies. The bottom-up generation of BFs is described in Section 3.

Software Features (SF) represent the high-level abstraction of an existing software system describing functional behavior or capabilities of an existing system. The different SFs are related to one another based on three different SF dependencies *decomposition*, *specialization*, according to [5], or *related to*. Figure 1 (b) illustrates a SF model excerpt of the software system developed by the company. In this example, the SF *System & Instruments* represents a decomposition of *Price Finding*, which means that *System & Instruments* refines the functional behavior of *Price Finding*. The *specialization* dependencies between *Negotiation* and SFs *Per Kit*, *Per Test* and *Per Reportable* indicate alternative functionalities of the SF *Negotiation*. The dependency *related to* indicates that the SF *Per Kit* and *Material List* are related to each other and a change to *Material List* has an impact on SF *Per Kit*.

Software Feature Deltas, as shown in Figure 1 (a), are those SFs that bundle one or more delta requirements, as for example *System & Instruments*.

Delta Requirements describe an enhancement to a specific SF [4]. We classify delta requirements using the delta requirement classification labels, described in Table 1. For example, in Figure 1 (a), delta requirement *instrument age in months* is linked with the SF delta *System & Instruments*. Since this delta comprises a specialization of an existing functionality, with the age of instruments managed in months instead of only years, the delta requirement is labeled with **S** for *specialization delta*.

Table 1. Delta requirement classification labels

Label	Description
Configuration delta	requires a configuration of the system that does not require an implementation of functionality.
Quality delta	addresses quality improvements, e.g. usability or performance.
Specialization delta	requires a specialization of existing functionality, with which the new and the existing functionality can be used optionally, e.g. filtering deals per day, week or also per month.
Enhancement delta	requires an enhancement of existing functionality.
Exclusive delta	requires a functionality that is conflicting with an existing functionality when behaving simultaneously.
Substitution delta	replaces existing functionality, where the old functionality is no longer available.

Applying RASM to structure SRP relevant information addresses the feature generation requirements FG1-FG4 and provides the following additional benefits. An *incremental documentation of a specific release* is provided, where business goals are related to solution specifications. Furthermore, a *business-oriented software feature-based representation of the whole system* is provided, which enables the comparison of different release versions based on any RASM meta-model element. For example, the following question can be answered: *What is the difference between release version X and Y with respect to SFs and corresponding delta requirements?* This can be a good basis to give a first estimation of the resulting end user training effort. Moreover, *SFs can be more appropriate to ask end users about SF usage or satisfaction*, because a SF model can be directly related to user manuals. Furthermore, the *identification of countries with similar markets*, based on their high priorities for the same BFs, can be easily supported. Additionally, *a measure for business strategy coverage* can be provided after feature selection is made, because the selected feature sets can be assessed concerning their strategy coverage. Finally, *the prioritization effort is reduced*, because the prioritization of low-level delta requirements is not necessary.

3 Bottom-up Feature Generation Approach

The major idea of the proposed approach is to handle continuously incoming low-level delta requirements at feature level to relate them more effectively to business strategies, where bug requests are not considered by the approach. These delta requirements are raised by the different company sites countries during support, test phases or roll-out projects and are bundled based on their SF belonging. Therefore, the approach includes the maintenance of RASM-relevant information in a bottom-up manner and assumes that an instance of the RASM exists. A RASM instance represents a specific increment (release) with respect to selected BFs and a SF model of the whole system. In the following the five steps of this approach are explained by describing goal and support of every step.

STEP1 Goal: ***The delta requirement is classified using delta requirement classification labels.*** Support: The classification of delta requirements is supported by the

given requirement classification catalogue in Table 1.

STEP2 Goal: ***The delta requirement is unambiguously related to one SF.*** The SF model-based representation of the existing system comprises ”decompose” and ”specialize” dependencies. If there are several potential SFs identified for a specific delta at the same level, the delta requirement is related to the parent SF. This indicates that a delta requirement potentially has an impact on all decomposed SFs below and the parent SF would be raised as a newly generated *software feature delta* in the model. For example, if the delta requirement *instrument age in months*, illustrated in Figure 1 (b), was identified as relevant to the SF *System & Instruments* and *KPI Calculation* the delta is related to *Price Finding*. Support: To support the identification of related SFs the requirement clustering approach according to [1] can be used to identify a relationship between the delta requirement under consideration and already bundled delta requirements.

STEP3 Goal: ***Delta requirement duplicates or conflicts are identified and removed.*** Support: Through the bundling of related delta requirements into the same SF group, duplicates are easier to detect. Additionally, the classification of delta requirements supports the recognition of conflicts between delta requirements.

STEP4 Goal: ***Software feature deltas (existing features that have to be changed) are identified, where these are the basis for release planning.*** Support: All SFs that were related to one or more delta requirements represent software feature deltas. Based on the classification of delta requirements, SF deltas can also be classified based on the deltas they bundle. For example, a SF delta comprising only delta requirements classified as configuration delta can be classified as a configuration SF delta, too.

STEP5 Goal A: ***The relation to the identified SF delta and a BF is assessed.*** Goal B: ***The relation to the identified SF delta and a business strategy is assessed and a new BF is generated.*** Support A and B: To identify a relation to a BF, already existing links between the SF, to which the delta requirement is related to, and BFs can be considered. Regarding the application example in Figure 1, the delta requirement *instrument age in months* is related to *System & Instruments*. We can see that there are already relations to two different BFs and it can be assessed whether the delta requirement is related to one of them or not. In the case of the application example, non of the existing BFs is sufficient, but a relation to the business strategy *Deal Compliance & Monitoring* is recognized. Therefore, a new BF *Deal Analysis* is generated that can be used for SRP purposes, e.g. gathering country priorities in order to assess the common priority.

4 Discussion

We have made some first experiences on generating a RASM instance at the company. The most challenging task was the generation of the SF model, representing the software system functionality at a high-level that enables to relate and further to understand delta requirements. Since, system solution specifications are only available incrementally from one release to another, user manuals and release notes are a good source to identify existing SFs and their relations. Once, a SF-based representation of the system exists, the proposed bottom-up feature generation approach maintains RASM by relating continuously arriving delta requirements to SFs and identifies SF deltas for release planning purposes. Furthermore, if SF deltas are recognized as business strategy

relevant, but cannot be related to existing BFs, a new BF is generated in a bottom-up manner. However, the top-down maintenance of the RASM, deriving BFs or SF deltas from business strategies, is an open issue.

Moreover, there are additional open issues that have to be discussed. It is not clear whether the SF-based representation of the system is detailed enough to understand delta requirements and to bundle them effectively. So far, the handling of delta requirements is not sufficiently addressed in requirements engineering research as stated by Herrmann et al. [4]. Further, it is not clear whether it is realistic that a suitable specification of the whole system is available in practice that can be used to sufficiently identify the belonging of delta requirements. Furthermore, we have not validated the suitability of the proposed delta requirement classifications to classify SF deltas. This requires, that SF deltas bundle a homogenous type of delta requirements, which is not always the case.

5 Conclusion and Future Work

In this paper we provided two contributions. First, we gave a more detailed explanation of the RASM, which copes with requirements and solution abstraction during SRP. Second, we provided a bottom-up feature generation approach to maintain the RASM and that helps to assess the business strategy relevance of continuously arriving delta requirements. Future work will include a case study in a large-scale industrial setting to evaluate the scaleability of RASM and the proposed bottom-up feature generation approach.

References

1. Chen, K, Zhang, W., Zhao, H.: An approach to constructing feature models based on requirements clustering. In: 13th IEEE International Conference on Requirements Engineering (RE05). pp. 31-40. (2005)
2. Godfrey, M.W., German, D.M.: The past, present, and future of software evolution. Frontiers of Software Maintenance (FoSM'08), pp. 129-138. (2008)
3. Gorschek, T., Wohlin, C. Requirements Abstraction Model. Journal of Requirements Engineering 11(1), pp. 79-101 (2005)
4. Herrmann, A.,Wallnoefer, A., Paech, B.: Specifying changes only - a case study on delta requirements. In: 15th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'09), pp. 45-58 (2009)
5. Kang, C. K., Cohen,S., Hess,J., Nowak, W., Peterson, S.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University. (1990)
6. Lehtola, L., Kauppinen, M., Kujala, S.: Linking the business view to requirements engineering: long-term product planning by roadmapping. In: 13th IEEE International Conference on Requirements Engineering (RE05). pp. 439-443. (2005)
7. Passos, L., Czarnecki, K., Apel, S., Wsowski, A., Kaestner, C., Guo, J.: Feature-oriented software evolution. In: 7th International Workshop on Variability Modelling of Software-intensive Systems (VaMoS 13). pp. 1-8 (2013)
8. Zorn-Pauli, G., Paech, B., Beck, T., Karey, H.: Analyzing an industrial strategic release planning process - a case study at Roche Diagnostics. In: 19th International Working Conference on Requirements Engineering: Foundation for Software Quality (2013) (accepted to appear)

Interactive Session: Cloud Computing Business Models – Trends and Implications for Software Product Managers

Hans-Bernd Kittlaus

InnoTivum Consulting
Im Sand 86, D 53619 Rheinbreitbach, Germany
hbk@innotivum.de

Abstract. In the Cloud Computing domain, business models are being invent ed, changed and reinvented more frequently than in any other industry. In this interactive session, we are going to look at the disaggregation of the value chain and the resulting business models of the players involved. The discussion will not only focus on the existing situation, but also on trends and their impacts for software product managers.

4 International Workshop on Requirements Prioritization and Communication (RePriCo)

Editors

Georg Herzwurm
University of Stuttgart, Germany, herzwurm@wi.uni-stuttgart.de

Wolfram Pietsch
Aachen University of Applied Sciences, pietsch@fh-aachen.de

Workshop Programme

Introducing RePriCo'13 103
Georg Herzwurm and Wolfram Pietsch

Position Paper: How Requirements Dependencies Make Requirements Prioritization Complex 107
Andrea Herrmann and Maya Daneva

Customer orientation in product development 115
Sandra Klute and Robert Refflinghaus

(Semi-)Automatic Categorization of Natural Language Requirements during Elicitation Research Preview 127
Daniel Ott and Eric Knauss

Guiding Requirements Elicitation using a Prioritization Framework 133
Norman Riegel

Introduction of RePriCo'13

Georg Herzwurm[1], Wolfram Pietsch[2]

[1] Department for Business Administration and Information Systems, esp. Business Software,
University of Stuttgart, Keplerstr. 17, 70174 Stuttgart, Germany
herzwurm@wius.bwi.uni-stuttgart.de
[2] Business Management, International Sales and Service Management
Aachen University of Applied Sciences, Eupener Str. 70, 52066 Aachen, Germany
pietsch@fh-aachen.de

1 Conception and workshop content

RePriCo'13 served as a platform for the presentation and discussion of new and innovative approaches to prioritization issues for requirements engineering with a focus on requirements communication. It represents the 4th International Workshop on Requirements Prioritization and Communication and has formerly been named International Workshop on Requirements Prioritization for customer oriented Software Development.

In order to address a broader audience we decided to slightly change the focus of RePriCo: the abbreviation kept the same but instead of customer oriented software development we want to further emphasize the communicational aspects in requirements engineering as a vital prerequisite for customer orientation. Coordination and negotiation of requirements between all stakeholders become more and more essentially for successful software development projects.

It is generally accepted that every software development project has to fulfill (at least some) customer needs to be successful. But unless you have unlimited resources like time and budget you have to decide which customer needs to fulfill (first) and which not (or later). Hence, requirements prioritization should be in most software development projects an inevitable activity.

The RePriCo workshops held at all previous REFSQ conferences each showed that prioritization is recognized as an important task within the requirements engineering process in order to cope with complexity and to establish focus properly. From a formal point of view it is merely a matter of choice of the right evaluation method and granularity of analysis. From a practical perspective it is also a matter of communication: consensus must be achieved about the appropriateness of requirements from the view of the customers and fed back into the process.

However, it is also an underestimated activity with little methodical guidance. Often only rough guidelines like "establish an absolute ordering", "take account of business value" or "discuss between all participants" are given.

In particular, the communication processes are negotiation processes dealing with the various requirements of all stakeholders and aiming at reaching an agreement. Customers and developers have a shared motivation in that they need each other to achieve the goal of reaching an agreement about the system to be developed.

Both need to communicate to exchange information – they strive for a consensus to reach a compromise decision ideally representing a win-win situation for all stakeholders.

We are glad about holding RePriCo'13 for the fourth time at REFSQ in Essen: in 2010 ambitious participants from research as well as industrial practice discussed two full research papers and four position papers in an open-minded and pleasant atmosphere; in 2011 four submissions were accepted as full research papers and one submission as short paper for the discussion during the workshop; in 2012 three full and two short papers had been discussed.

RePriCo'13 attracted six submissions. Each submission was reviewed by three experts of the program committee (chairs and/or members). The members of the organizing committee assigned reviewers to each submission depending on the research and practical background of each reviewer matching to the title and abstract of each submission. To avoid any conflict of interest the organizing committee took care of not to assign more than one reviewer to a submission who might know one of the authors of a submission personally. To identify excellent papers first the rating scale within the conference system EasyChair was used: an overall rating by each of the three reviewers weighted by the reviewer's confidence led to a ranking of all submissions. Secondly, subject to time and slots available for a half-day workshop depending on the length of a submission, the chairs of the program committee turned the balance to accept or to reject a submission. Therefore especially the matching of a submission to the workshop topics was taken into account. Finally, two submissions were accepted as full research papers and two submissions as short respectively position papers.

The submissions comprise current research findings and previews from various fields: requirements dependencies and their effect on complexity of requirements prioritization (Andrea Herrmann, Maya Daneva); development of a 10-dimensional model for structuring of requirements (Sandra Klute, Robert Refflinghaus); discussion of natural language requirements during requirements elicitation (Daniel Ott, Eric Knauss); usage of a business-process-driven software development framework (Norman Riegel).

Results of our workshop evaluation (questionnaires filled out by all attendees) showed, apart from a positive overall evaluation of the workshop, that the variety of research findings and the following discussions pleased all participants.

We are convinced that the workshop was rewarding like in previous years and findings in these proceedings encourage researches as well as software developers, requirements engineers or consultants to absorb new ideas and carry them out into their daily work and research projects.

Our special thanks go to all speakers and participants for their contributions to the workshop. Additionally, we would like to thank Camille Salinesi and Raúl Mazo as REFSQ2013 workshops chairs and Tobias Kaufmann and Stella Roxana Klippert as REFSQ2013 organizing team for their professional support. Last but not least we thank Benedikt Krams and Sixten Schockert for their effort in organizing RePriCo'13. We are confident in hosting RePriCo in 2014 once more and are looking forward to welcoming many participants again.

2 Organization

2.1 Program Committee

Chair.
Prof. Dr. Georg Herzwurm, University of Stuttgart, Germany
Prof. Dr. Wolfram Pietsch, Aachen University of Applied Sciences, Germany

Member.
Dipl.-Math. Peter Brandenburg, Vodafone D2 GmbH, Germany
Dr. sci. Math. Thomas Fehlmann, Euro Project Office AG, Switzerland
Dr. Andreas Helferich, highQ Computerlösungen GmbH, Germany
Priv. Doz. Dr. Andrea Herrmann, Herrmann & Ehrlich, Germany
Prof. Dr. Thomas Lager, Grenoble Ecole de Management, France
Dipl.-Betriebswirt (FH) Olaf Mackert, SAP AG, Germany
Dipl. Wirt.-Ing. Waldemar Meinzer, Volkswagen AG, Germany
Priv. Doz. Dr.-Ing. Robert Refflinghaus, TU Dortmund University, Germany
Prof. Björn Regnell, Lund University, Sweden
Dipl.-Wirt.-Inf. Sixten Schockert, Universität Stuttgart, Germany
Prof. Dr. Hisakazu Shindo, University of Yamanashi, Japan
Dipl.-Ing. Gerd Streckfuß, iqm Institut für Qualitätsmanagement, Germany
Prof. Dr. Yoshimichi Watanabe, University of Yamanashi, Japan

2.2 Organizing Committee

Dipl.-Kfm. (FH) Benedikt Krams, M.Sc., University of Stuttgart, Germany
Dipl.-Wirt.-Inf. Sixten Schockert, University of Stuttgart, Germany

Position Paper: How Requirements Dependencies Make Requirements Prioritization Complex

Andrea Herrmann[1], Maya Daneva[2]

[1]Free Software Engineering Trainer and Researcher, Stuttgart, Germany,
herrmann@herrmann-ehrlich.de
[2]University of Twente, the Netherlands, m.daneva@utwente.nl

Abstract. [Context and motivation] Requirements prioritization in practice is still a challenge. **[Question/problem]** This task takes a lot of time, and the resulting priorities are not always optimal. **[Principal ideas/results]** This position paper discusses the complexity of the requirements prioritization task. **[Contribution]** New light is shed on requirements prioritization methods, by taking a complexity perspective on them.

Keywords: prioritization, requirements prioritization, task, task complexity

1 Introduction and Motivation

Prioritizing requirements is a complex activity, not only due to high volumes of requirements, but also due to the trade-offs among several criteria, like requirements' cost and benefit. This task is complex also due to dependencies among requirements, respectively among their cost and benefits. The usefulness of some requirements may well be contingent on others, and some requirements may be implementable only if others are realized first. Another dimension of the problem is that the prioritization criteria depend on the perspective of each stakeholder or on subjective perceptions. E.g., cost can mean the implementation cost experienced by the development team, but can also include development cost caused in the environment (e.g. preparation of a technical landscape for the new system, including technical interfaces), roll-out cost (e.g. training or cost of downtime of the system) or maintenance cost provided by the administrator. Usually, not all of them are included in the prioritization phase. Other criteria like benefits depend even more on the perspective chosen, as not all stakeholders profit equally of each requirement.

Requirements prioritization methods reduce the complexity of the prioritization task by focusing on the most important criteria and breaking down the complex activity into simpler steps. These simplifications mean to focus on some aspects and disregard others. In what follows, Sect. 2 discusses four factors making requirements prioritization complex and Sect. 3 – how prioritization methods reduce the complexity of this task. Sect. 4 summarizes the paper.

2 Complexity of Requirements Prioritization

According to Brooks [4], complexity in software engineering is the sum of the essential complexity of the problem plus accidental complexity. Only the accidental complexity is caused by the way in which the problem is presented and solved; the essential complexity is inherent to the problem. The essential complexity is caused by both a high number of elements and dependencies among them. Essential complexity is also caused by external requirements and external interfaces, and by change. This view is shared by researchers in social sciences (e.g. in organizational behavior) who study human activities. For example, Wood [23] defines an objective measure of task complexity as the weighted sum of the number of components (component complexity), the degree of interaction between the components (coordinative complexity) and the degree of temporal change of the relation between task-related input and output (dynamic complexity). These four factors of (essential) task complexity are at play in the process of prioritizing requirements:

a) Assuming, there are hundreds or thousands of requirements in one project, the number of requirements to be prioritized can be reduced by means of a pre-selection. Requirements triage [5], in analogy to the treatment of a disaster´s victims by medical personnel, distinguishes among three types of requirements, and only one of these needs more detailed prioritization. For the other two, it is clear that they either will be implemented in the next release or will not.
b) The requirements and their priorities depend on each other. If two requirements are only useful when implemented together, postponing one of them to a later release, reduces the other requirement´s priority. Or if the stakeholders need at least one out of two requirements, postponing the one raises the other requirement´s priority. If the priority depends on cost, then technical dependencies between requirements play a role. If implementing a requirement makes it easier, and consequently cheaper, to implement another, the priority of the second requirement will raise once there is a decision to implement the first. Also, requirements specifications from different perspectives depend on each other, e.g. customer requirements and technical requirements. Requirements dependencies are discussed further below.
c) There are external requirements, e.g. a fixed budget, and external constraints of the requirements prioritization task. E.g. the perspectives of the diverse stakeholders to be considered. Each stakeholder group might have their own prioritization criteria (e.g. one may want to get benefit out of the system, another − business value or market value, while a third one may want to reduce risk). If the objective is to calculate one priority value for each requirement, these criteria must somehow be balanced against each other. E.g., risk can be expressed as cost [24] or subtracted from the benefit [1],[6].
d) Dynamics: Requirements priorities change over time, just like requirements do [7], [16].

Dependencies among requirements: The benefits realized by individual requirements depend on each other. This is even true when all requirements are described from the same perspective (e.g. customer versus technical). An example of such dependencies is the following: Assuming a project to implement an email application, requirement A is "writing outgoing emails" and requirement C is "reading incoming emails". An email user left with only one of these two functionalities will consider the application of low value, because for an email exchange, both are necessary together.

For discussing requirements dependencies and their effect on benefits and on benefit estimation, we use the following generic model borrowed from Utility Theory [15], [8], Decision Theory and Mathematical Economics [22]. We assume that the following process of reasoning is analogous for cost, risk, and any prioritization criterion.

Let a Benefit Function B(S) model the benefit provided by an IT system S in which a certain number out of N candidate requirements is realized, while others are not [10]. We assume B(S) describes the system´s benefit completely. It can be quantified in a financial unit like € but also in relative values, e.g. on a point scale. One of the challenges during requirements prioritization is that the benefit is not known and cannot be measured in early development phases because the benefit will be only realized after the implementation, during the system´s usage. However, requirements prioritization usually means to make decisions before the implementation. Therefore, one needs an estimation predicting the future benefit as well as possible.

Assuming there is a set of N candidate requirements on S. S is then defined by those of the N requirements that will be realized or not (or partially). B(S) depends on S only, not on its history. We note these are general assumptions without unnecessary restrictions.

We can visualize this Benefit Function like this: In a two-dimensional space, the x-axis means the implementation degree x of requirement A. $\boldsymbol{x}$ can take values of 0 or 1. The y-axis shows the implementation degree of requirement C. If only these two (N=2) requirements exist and are either implemented or not, then the system S has four possible configurations: (I) Both requirements are not implemented (in S). (II) Requirement A is implemented, but not requirement C (S_A). (III) Vice versa, A is not implemented, but C is (S_C). (IV) Both requirements are implemented (S_{AC}). B(S) is completely known, if it is known for all four systems. Potentially, the implementation degree of a requirement can also take intermediate values like 0.3 or 0.5, especially for non-functional requirements. Then, the Benefit Function depends non-linearly on the requirement´s implementation degree x (see in the QUPER model [17]).

The form of B(S) models all requirements dependencies completely. For instance, when A and B need each other, then $B(S_A)$ and $B(S_C)$ are not much higher than B(S), but $B(S_{AC})$ is. When A and B could replace each other, then $B(S_A)$ and $B(S_C)$ are almost as high as $B(S_{AC})$.

The benefit of a single requirement A can be defined as $b_A(S) = B(S_A) - B(S)$, i.e. the benefit gain when adding requirement A to system S. This value depends on S; that is, it is different for different systems S. For instance, $B(S_A) - B(S)$ is not equal to $B(S_{AC}) - B(S_C)$. Also, the benefit is not additive, i.e. the benefit of a group of

requirements is not the sum of the benefits of the requirements. This 'non-additivity' means, that benefit estimation for a single requirement only makes sense relative to a clearly defined system S, which we call the reference system. In order to achieve benefit estimates which are comparable to each other, one usually estimates all requirement benefits relative to the same reference system. This reference system can be the status quo, the situation without software support, the mandatory requirements or the perfect system where all requirements are assumed to be implemented.

When implementing a requirement adds a certain benefit, then not implementing it can be expected to cause a dissatisfaction of the same value. However, several authors find that satisfaction and dissatisfaction are not equal [18], [20]. [10] explains how this results from the dependencies among requirements. This effect happens when benefit and dissatisfaction are not estimated with respect to the same reference system. For instance, when two requirements A and C need each other, the implementation of each of them adds only a low benefit, although it is high when implementing both. When using S_{AC} as a reference and estimating the dissatisfaction if one of them is not implemented, both requirements seem to be more important than with respect to the reference system S. In decision theory, this effect is known as the anchoring effect: Decision outcomes are compared with the reference system and this may affect the decision result.

The Benefit Function model can also explain the difference between an excitement requirement and a basic requirement, according to Kano´s terminology [11]. An excitement requirement when realized causes a high benefit, but when not realized, no-one will miss it. A basic requirement when being implemented is not perceived to add any benefit, but its absence will cause high dissatisfaction. We note that both types of requirements might be equally beneficial, however, the perceived difference between them is due to the fact that the requirements specialist tacitly assumes the state of the art or market to be the reference system for the prioritization process. In this reference system, the basic requirement is a default, the excitement factor is not. In turn, adding the excitement requirement to the reference system adds perceived benefit, while the basic requirement being part of the reference system, cannot add benefit, but only can cause dissatisfaction if left out.

When $B(S)$ is completely known (i.e., for all possible S), all requirement benefits $b_A(S)$ (for all systems S and requirements A) are defined and can be determined from $B(S)$. Attributing one fixed benefit value to each requirement disregards dependencies. However, this is what all of the state-of-the-art requirements prioritization methods do [9]. They - only in an implicit and approximate fashion if at all - take into account that the benefits of requirements depend on each other. Only one recent work undertakes to consider requirements dependencies explicitly [14].

3 How prioritization methods reduce the complexity of requirements prioritization

Usually, *B*(S) is unknown. The complete estimation of B(S) is practically impossible. For estimating the Benefit Function for the complete space of possible systems S built with *N* requirements would mean to estimate the benefit for at least 2^N systems. This means that for only *N*=10 requirements, 1024 systems S exist for which *B*(S) must be estimated. If the estimation of one value demands only one minute, then 17 hours would be needed for estimating the complete Benefit Function.

Consequently, one cannot expect the estimators to determine the complete function *B*(S). Prioritization methods organize the prioritization task in a way to reduce complexity. In [9], we developed a framework describing how requirements prioritization methods treat requirements dependencies and categorized well-known methods according to this framework. In this paper, we add a discussion of how each simplification reduces complexity. We distinguish six ways of treating dependencies:

1) Each requirement´s benefit (or priority, or cost) is assumed to be a fixed value: This approach disregards all dependencies among requirements and allows to treat each requirement individually. This is commonly done by state of the art requirements prioritization methods [9]. However, this is a strong oversimplification when requirements dependencies are important, and it only makes sense when the reference system is the current state of the system and only few new requirements are to be considered, like in agile development or software maintenance.

2) Grouping requirements: Requirements are grouped into bundles in a way that each group is relatively independent of the others. This grouping takes care of the most important dependencies and disregards all others. The groups can be built on different levels to form a hierarchy of requirements [13], which reduces the complexity of the estimation task by first prioritizing the groups relative to each other and then, if needed, the individual requirements within a group (as in [19]). What is more, the grouping reduces complexity by reducing the number of requirements to be considered.

3) Pair-wise comparison: Pair-wise comparisons reduce estimations to comparing only two requirements with each other, what is cognitively easier. However, pair-wise comparison methods assume that each requirement has a fixed benefit, which in turn means to disregard dependencies.

4) Modelling pair-wise requirements dependencies: The Analytic Network Process ANP [21] method represents pair-wise requirements dependencies in a “supermatrix” and includes them in the calculation of the requirement benefits from the results of pair-wise comparisons. Requirements dependencies might be more complex than pair-wise relationships, but the supermatrix allows to model the most important dependencies.

5*) Using discrete values instead of a continuous scale:* This means using a set of categories, e.g. a nominal scale like the values 1-2-3, or low/ medium/ high, or mandatory/ desirable/ inessential [12], [2]. Such decisions are easier to make and it is not necessary to compare each requirements with each other. These priority categories allow only coarse-grained decisions with respect to requirements.

6) Estimating benefit intervals: Some authors advocate that intervals be used, e.g. by estimating an optimistic, realistic and pessimistic value [5], [3]. These can make transparent uncertainties and the influence of information not yet available.

Treating a requirement´s benefit as a fixed value and doing pair-wise comparison are the worst of all solutions, because they neglect dependencies completely. Using discrete values and estimating intervals are second worst. These approaches accept that there are uncertainties. Intervals quantify their size individually per requirement, while using discrete values defines an approximate size of uncertainty for all requirements. However, different types of uncertainty are included, not only the influence of dependencies and this makes the estimation results very coarse-grained. The best solutions are 'grouping requirements' and 'modelling pair-wise requirements' dependencies because they explicitly document and consider the most important dependencies and neglect only less important ones.

4 Summary

This paper discusses why and how requirements prioritization is a complex task. It seems evident that this complexity must be reduced in one way or another. We, then, present six ways that are commonly used in state-of-the-art requirements prioritization methods to reduce complexity. We also discuss the strength and weaknesses of how each one in reducing complexity.

5 References

1. Azar, J., R.K. Smith, D. Cordes, Value Oriented Prioritization: A Framework for Providing Visibility and Decision Support in the Requirements Engineering Process, TR, Dept. of CS, University of Alabama, 2006.
2. Beck, K., Extreme programming explained, Addison Wesley, Upper Saddle River, USA, 2000.
3. Boehm, B.W., R.E. Fairley, Software Estimation Perspectives, IEEE Software, Nov/Dec 2000, pp.22 26.
4. Brooks, F.P., No Silver Bullet: Essence and Accidents of Software Engineering, Computer, Vol. 20, No. 4, April 1987, pp. 10 19.
5. Davis, A.M., The Art of Requirements Triage, IEEE Computer 36(3) March 2003, pp. 42 49.
6. Denne, M., J. Cleland Huang, Software by Numbers: Low Risk, High Return Development. Prentice Hall, 2003.
7. Ebert, C., J. De Man, Requirements Uncertainty: Influencing Factors and Concrete Improvements, Proceedings of ICSE 2005, pp. 553 560.
8. Fishburn, P.C., Utility Theory for Decision Making. Robert E. Krieger Publishing Co, Huntington, USA, 1970.
9. Herrmann, A., M. Daneva, Requirements Prioritization Based on Benefit and Cost Prediction: An Agenda for Future Research, 16th IEEE RE 2008, pp.125 134.
10. Herrmann, A., B. Paech, Benefit Estimation of Requirements Based on a Utility Function, 12th REFSQ (2006), Essener Informatik Beiträge, Band 11, pp. 249 250.
11. Kano, N., Attractive Quality and Must be Quality, Journal of the Japanese Society for Quality Control, H. 4, 1984, pp. 39 48.

12. Karlsson, J., Software requirements prioritization, Proc. 2nd IEEE RE'1996, pp.110 116.
13. Karlsson, J., S. Olsson, K. Ryan, Improved Practical Support for Large scale Requirements Prioritisation, REJ 2(1) 1997, pp.51 60.
14. Kolbe, C., Requirements Prioritization by Using Requirements Relations. Workshop Requirements Prioritization for Customer Oriented Software Development (RePriCo), REFSQ 2012 Workshop Proceedings, pp. 139 152.
15. Neumann, J. von, O. Morgenstern, Theory of Games and Economic Behavior. Princeton University Press, Princeton, NJ, 1944.
16. Pena, M., R. Valerdi, Characterizing the Impact of Requirements Volatility on Systems Engineering Effort, 25th Int. Forum on COCOMO and Systems/Software Cost Modeling, USC, LA, CA, Nov 2 4, 2010.
17. Regnell, R., M. Höst, R.B. Svensson, A Quality Performance Model for Cost Benefit Analysis of Non Functional Requirements Applied to the Mobile Handset Domain, 13th REFSQ'07, 2007.
18. Robertson, S., J. Robertson, Mastering the Requirements process, Addison Wesley, 1999.
19. Ruhe, G., A. Eberlein, D. Pfahl, Trade Off Analysis for Requirements Selection, IJ of SEKE 13(4) 2003, pp. 345 366.
20. Rupp, C., Requirements Engineering und Management Professionelle, iterative Anforderungsanalyse für die Praxis, Carl Hanser Verlag, 2004.
21. Saaty, T.L., The Analytic Network Process: Decision Making with Dependence and Feedback, RWS Publications, 2005.
22. Schofield, N., Mathematical Methods in Economics and Social Choice. Springer, 2002.
23. Wood, R., Task complexity: Definition of the construct. Organizational Behavior and Human Decision Processes 37, 1986, pp. 60 82.
24. Xie, N., N.R. Mead, P. Chen, M. Dean, L. Lopez, D. Ojoko Adams, H. Osman, SQUARE Project: Cost/Benefit Analysis Framework for Information Security Improvement Projects in Small Companies, Technical Note CMU/SEI 2004 TN 045, Software Engineering Institute, Carnegie Mellon University, 2004.

Customer orientation in product development

Sandra Klute[1], Robert Refflinghaus[2]

[1] RIF e.V. ,
Joseph von Fraunhofer Str. 20,
44227 Dortmund, Germany
Sandra.klute@rif ev.de
[2] RIF e.V.,
Joseph von Fraunhofer Str. 20,
44227 Dortmund, Germany

Abstract. Continuously winning new customers demands arousing customers' enthusiasm. Therefore, it is necessary to deal with customers' needs. In this context, a comprehensive requirements management which includes identify ing, gathering and implementing requirements is essential. When developing complex products a multitude of divergent requirements of different stakehold ers occurs and has to be gathered and managed. Moreover, the requirements are of different importance, different level of specification and often are not inde pendent of each other. They rather interact or interfere. Hence, structuring of requirements is necessary. For this purpose, a 10 dimensional model has been developed within a German collaborative research centre. This model is pre sented in the paper.

Keywords: requirements, structuring, product development

1 Introduction

Customer orientation has become an essential aspect for companies to maintain competitiveness. In this context, the focus of quality management has shifted from quality control to the early phases of the quality loop. Especially, dealing with the stakeholders' needs and wishes is a task of paramount importance for an adequate planning and developing of products. Therefore, it is necessary to establish a systematic and holistic requirements management which comprises gathering, structuring and implementing requirements into product features. Thereby, a necessary premise is the identification of all stakeholders. Neglecting requirements or integrating them after the early phases of the product development might entail high costs or even is not possible. Hence, this should be avoided. In the case of industrial buying (B2B) this aspect is particulary important, especially when planning complex products like for example intra-logistical facilities or software. This results from the great amount of possible stakeholders with different and often divergent requirements. So there might be conflicting requirements. Moreover, there are often problems regarding the meaning and content of the requirements although the stakeholders exchange information. This is because the stakeholders have different professional and functional background and so for example use different terms but mean the same thing or have a divergent under-

standing of a special term. Beyond that, an adequate and systematic requirements management is difficult because of the multitude of requirements which have to be managed. Additionally, it has to be considered, that the requirements may be very different from each other or even conflictive due to the different aims of the stakeholders. In summary, the aspects lead to inconsistency and uncertainty in the planning process. Therefore, existing knowledge and experience from former projects are often the base for decisions instead of stakeholder requirements.

The established quality management method quality function deployment (QFD) can be applied for transforming requirements into product characteristics. Nevertheless, in general this method presumes a requirements' structuring respectively grouping in advance to be able to handle the multitude of information. In addition, the often different level of specification and existing dependencies between requirements have to be considered to receive sensible results [1]. In this context, structuring models can be used for grouping the requirements. Nonetheless, a data processing is necessary to manage and provide the great amount of information [2].

Thus, an interdisciplinary cooperation of the domains or rather departments of quality management, marketing and information technology is required (c.f. figure 1).

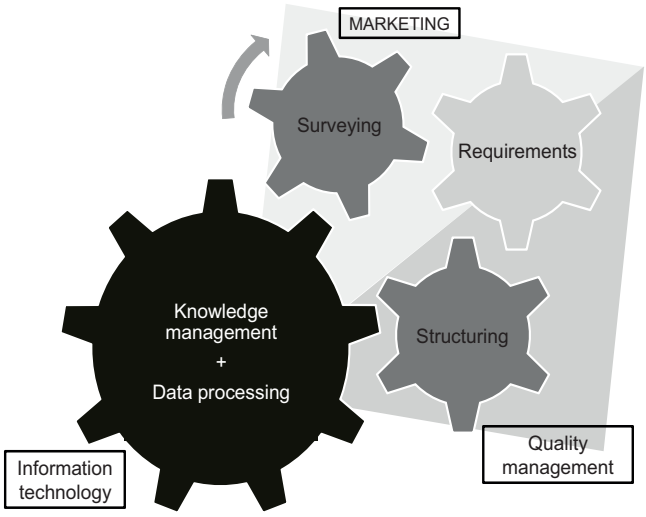

Fig. 1. Cooperation of different disciplines for a comprehensive requirements management [3]

2 Importance of the Identification of Stakeholders and their Requirements

Stakeholders are groups which are essential for a company's survival and which are related to or impacted by corporate activities [4]. They can be further divided into internal and external ones. The first group, the internal stakeholders, comprises the company's staff, and can be further differentiated into the company's departments, like for example, the purchasing department and the management which (might) have divergent requirements. External stakeholders are customer, but also residents, that live near the company, and the law making, for instance, regulations regarding noise disturbance. All of these stakeholders and their requirements have to be taken into account during the planning and developing process.

Thereby, the requirements are often not independent from each other, but rather are supporting or conflicting due to the divergent interests of the stakeholders. In addition, during the planning requirements dealing with all phases of the life cycle of a product, including disposal and reuse, have to be regarded. Especially, the topic sustainability and going along with it requirements dealing with eco-friendliness, low energy and material consumption and recycling have received greater interest in the last years. For buying decisions of the customers they have become essential [5]. Neglecting of stakeholders or their requirements may cause severe consequences, especially if the importance of the stakeholder is high or if a requirement is of high relevance for a stakeholder. To identify and consider all stakeholder and their requirements during the planning and developing process it is mandatory having an adequate structuring model which allows all requirements to be surveyed and displayed. Also, the model should help to receive an overview of a potential lack of information if for example stakeholders have been disregarded. This might be the case if for some categories or dimensions no requirements have been gathered.

3 Requirements' Fulfilment - Base of an holistic Requirements Management

Literature offers a multitude of structuring methods [6,7,8]. Nevertheless, most of the methods only deal with the structuring of gathered requirements. They do not consider that the fulfillment of the requirements is essential for the customer satisfaction which again is important for winning over the customer. So, they do not offer a comprehensive view including feedback and customer satisfaction. The aspect that requirements' fulfillment is important for customer satisfaction and thus for a long-term customer relationship is not regarded by the methods and hence during the planning process. Though, integrating stakeholders' evaluations of requirements fulfillment into the planning process contributes to achieve customer-orientation.

However, the obvious quality of a product for a customer is essential and influences the customer satisfaction. Thereby, the customer satisfaction results from the difference between nominal condition, i.e. the requirements, and actual condition, i.e. the requirements' fulfilment. Thereby, it has to be checked if and to which degree the requirements are fulfilled. In this context, the subjective and objective evaluation has to be differentiated because it is not possible to measure the fulfillment of every requirement objectively. There are requirements which can only evaluated by humans and not by measurement devices like for example "attractive design".

For the subjective evaluation senses, feelings and the case history are important. For evaluating services additionally, the ServQual-approach has to be regarded. This approach evaluates based on the factors assurance, reliability, tangibles, empathy and responsiveness [9]. Hence a holistic requirements management should include these aspects. By this, the importance of regarding the whole quality loop instead of single phases becomes clearer.

Moreover, the importance of a requirement for a stakeholder, i.e. the requirement's weighting, has to be considered, too, because not all requirements are equally important for the stakeholder. This also affects the customer satisfaction. When analysing this relation, the KANO-model can be used. Thereby, the allocation of requirements to the different Kano categories is linked with the influence of individual product characteristics on the customer satisfaction.

Within the KANO-model the following requirement categories are differentiated: [10]:

1 must be or basic requirements (M): If a product feature of this category is not available or the performance of this product feature is low, customers' dissatisfaction increases. However, high performance of basic requirements does not raise customer satisfaction above a neutral level.
2 one-dimensional or performance requirements (O): With this, a linear correlation between customer satisfaction and the performance of the corresponding product characteristic is assumed. Low customer satisfaction leads to low attribute performance and vice versa.
3 attractive (delight and surprise) requirements (A): Attractive requirements are not explicitly expressed or expected by the customer. Hence, fulfilling these requirements entails disproportionate satisfaction, but not fulfilling attractive requirements does not cause feelings of dissatisfaction.
4 indifferent requirements (I): The performance of the product features does not have an influence on customer satisfaction.

Nevertheless, it is necessary to modify this model for being able to consider the weighted requirements and the weighted level of satisfaction. For the stakeholders each requirement may have a different weighting, whereas within the Kano model only three or four groups of requirements are distinguished. Although, by integrating a weighting factor, an exact depicting of graphs of requirements is not possible, because the graphs may have a flatter or steeper progress.

4 10-Dimensional Structuring Model

This structuring model has been developed because considering only requirements and disregarding their fulfillment is insufficient. Moreover, one should regard not solely the "product core", i.e. the product's components or features for fulfilling the functions stakeholders' needs [11], but also services and delivery, as they are of increasing importance for the customers and their buying decisions and a lot of stakeholder requirements deal with aspects going along with them [12]. Hence, the extended product should be the reference object to which the requirements refer instead of only the product core. Thereby, the reference object might be a product or a process. Actually, a holistic approach is essential. This approach should include the stakeholders' evaluation of the requirements' fulfillment as well as the customer satisfaction which results from it instead of only the regarded product respectively the requirements made on it. This enables to give a feedback between requirements and their fulfillment. Additionally, it enables to consider latent requirements, i.e. requirements the stakeholders have, but do not mention, which otherwise are often neglected (Figure 2). So, during the different stages of the planning process it can be assessed if and to which extent the product or process being developed meets the requirements. The received results then can be included in the further development process.

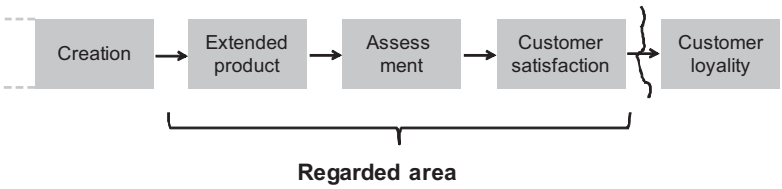

Fig. 2. Basic idea of the model.

The ten dimensions of the model are: obligations, surroundings, information, economy, qualification, technical-functional requirements, product, weighted level of performance and customer satisfaction.

As they serve different purposes those dimensions differ from each other concerning their content and meaning. Whereas the first six mentioned dimensions, disregarding the dimension time, serve to structure the requirements, i. e. the nominal condition, the dimension weighted level of performance serves to measure the reaction of the stakeholders with respect to the fulfillment of their requirements by comparing nominal and actual condition. The satisfaction of the stakeholders resulting from their evaluation of the requirements' fulfillment is shown in the dimension customer satisfaction.

Thus, the customers' reaction to existing attributes of the (real) product or within the framework of tests, like for example three-dimensional simulations has to be surveyed. Comparing nominal and actual condition shows whether and to which extent the stakeholder requirements are fulfilled. As a result, the dimensions "weighted level of performance" and "customer satisfaction" capture the actual condition and are

temporally behind the other ones, whereby the customer satisfaction results from the requirements' fulfillment and the importance of a requirement and hence its fulfillment for the customer. The dimension product serves to structure the reference object to which the requirements refer. Hence, it does not comprise requirements in their classical meaning.

Table 1. 10 dimensional model.

Structuring of requirements with regard to content	Structuring of the reference object	Evaluation of requirements' fulfillment	Temporal structuring of requirements
• Obligations • Surroundings • Economy • Information • Qualification • Technical-functional requirements	• Product or process (Reference object)	• Weighted level of performance • Customer satisfaction	• Time

In addition, time aspects have to be regarded. In this context, it has been considered that requirements are dynamical, that means they change over time regarding their meaning, their content and their level of specification, and that they are of different importance in different stages of life cycle. For example, stakeholders may not be able to articulate all of their requirements at the beginning of the planning process and may not be precise in their requirements [13]. Hence, the aspect of time has been considered as a separate dimension since this is essential for adequate product design and service over the whole life cycle. However, this dimension is not independent or comparable dimension to the others but it indicates which phase of the life cycle a requirement deals with and when it is mentioned.

To adequately structure and provide a topical classification of the requirements, the dimensions have to be further detailed into different categories. Initially, the model has been developed for the fields of intralogistics and with it the development of complex facilities. However, it is possible to apply and extend it to other fields as it is generic. This can be done by adapting the dimensions and their categories if necessary. Thereby, all dimensions serving to structure requirements can be separated into several categories and sub-categories. So, an adequate matching of the requirements which considers the requirements' level of specification can be done. With it, laminations should be avoided so that a valid interpretation by comparing these categories is enabled. As a result, categories of different dimensions should not be too analogous [5]. However, it is inevitable that they deal with the same topic but from a different point of view and with different focus. For example, the aspect of environment is

treated within the dimensions obligations and surroundings. The dimension "obligations" thereby deals with requirements which regard environmental protection by observing laws and provisions concerning this matter. Whereas the dimension "surroundings" deals with aspects which are not necessarily dealt with by law like for instance stakeholder requirements concerning radiation and sonic going beyond legal requirements.

Furthermore, it should be taken into account, that requirements should be classified in the space which is spanned by the ten dimensions although the requirements should not necessarily have to be sorted into all dimensions. The dimensions allow to determine and to structure all requirements on intra-logistical facilities in the case at hand and complex products or processes in general. Therefore, a further detailing and categorizing of the dimensions is requisite to ensure a topical classification of the requirements taking into account their level of specification.

In the following, the dimensions which serve to structure the requirements the dimensions surroundings, information and technical-functional are exemplarily presented in more detail. Additionally, the dimensions weighted level of performance and customer satisfaction are described in more detail as they give feedback to the requirements gathered.

4.1 Structuring Requirements according to the Dimension "Surroundings"

One dimension for structuring requirements is the dimension surroundings. Not the reference object itself, like for instance an intra-logistical facility, belongs to this dimension, but requirements concerning the facility's surroundings. This can be subdivided into the categories direct facility surroundings, resources, environment and safety.

The category surroundings comprises requirements concerning the direct surroundings of the facility. It includes for instance the facility's area, collaboration with other facilities and the interference factors. The latter are devices or components whose operating or existence near the facility bear (negative) consequences for the intra-logistical facility.

Personnel as well as material resources which are needed for manufacturing and operating the facility belong to the category resources. The personnel resources may be of qualitative as well as of quantitative kind. That means that this category includes requirements which result from the number and the qualification of the personnel that is available. Requirements concerning material resources can be differentiated by the kind of resource into for instance power, gas and water.

The category environment can be divided into macro-environment and micro-environment, depending on whether the requirements are of global kind or just concerning the close facility-environment. The first category includes requirements of social, political or technological kind. Hence, it refers to factors which cannot be influenced by the company directly. Requirements concerning the micro-environment are focused at the direct facility-environment. They usually can be influenced by the company. Aspects like radiation, sonic and so on belong to this category.

The fourth category in this dimension includes requirements dealing with the facility's security. Thereby, the aspect of occupational safety is very important. Re-

ments referring to this aim are workplace-design and the facility's construction. For this, the facility's dangerous spots are to be marked respectively to be secured to reduce or avoid the danger of an accident or an injury. The dimension "surroundings" is summarized and depicted in the following figure.

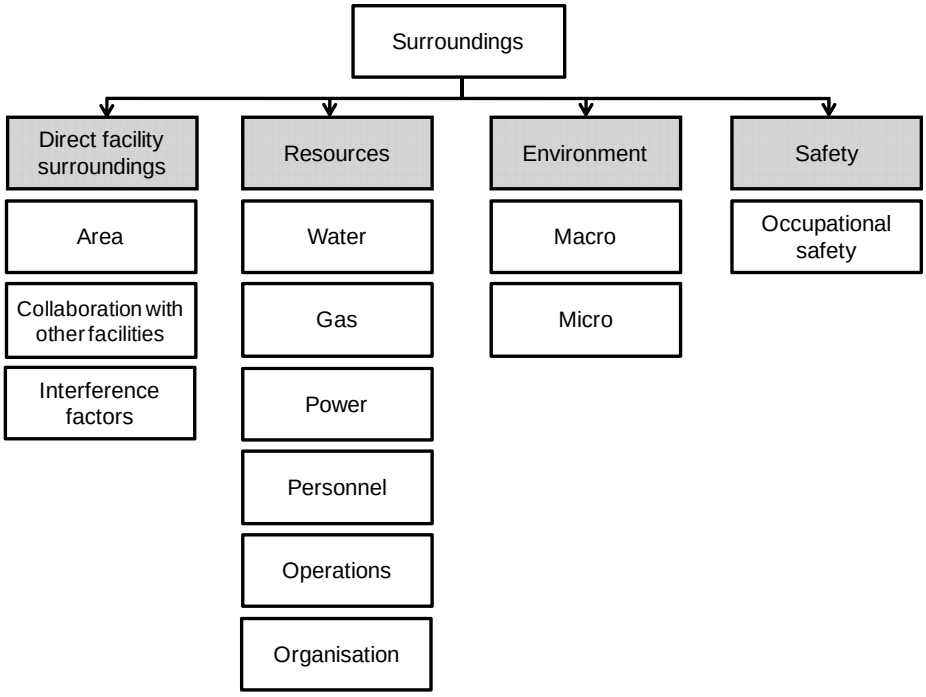

Fig. 3. Structuring Requirements according to the Dimension "Surroundings"

4.2 Structuring Requirements according to the Dimension "Technical-functional Requirements"

Requirements can also be of a more general kind as they do not concern directly individual parts of the regarded product. In fact, they concern the whole product and/or its function capability and performance like for instance low fault liability and low energy consumption. Hence, for considering these requirements one dimension of the model for structuring requirements is the dimension "technical-functional requirements". This dimension can be further differentiated into the categories "function-specific" and "supportive requirements". Thereby, within the category, "function-specific requirements" a further differentiation into the sub-categories reliability, functionality, robustness, load capacity and flexibility can be made. In this context, the aspect of reliability is of high importance. Reliability is defined as the ability of a reference object to perform its functions during a period of time [14]. To reach a higher level of specification, it can be measured for instance by using "mean time to failure", "mean time between failure" and "failure rate" [15].

The category “supportive requirements” includes requirements dealing with aspects which are not part of the core functions of the facility. It can be sub-divided into the sub-categories for “planability”, “producibility”, “installability”, “maintainability”, “operability”, “removability”, “recyclability” and “disposability”. “The parts of the facility should be removable easily.” Or “The product should be able to be disposed eco-friendly” for instance are requirements belonging into this category. Thereby, the first named requirement has to be sorted into the sub-category “removability” and the latter one into the category “disposability” (Figure 5).

Moreover, in this context the factor of maintainability has become of high importance for customers, especially for complex products having longevity. Hence, many requirements deal with this topic. Thereby, maintainability is defined as the condition of an entity concerning its appropriateness for maintenance. The requirement “The facility should feature a good maintainability.” could be mentioned as an example for a requirement on the maintainability. The maintenance includes measures to maintain or if necessary to return to the required functional status. Furthermore, maintenance can be further differentiated into the sub-groups “preventive maintainability”, “inspectability” and “reparability”. [16].

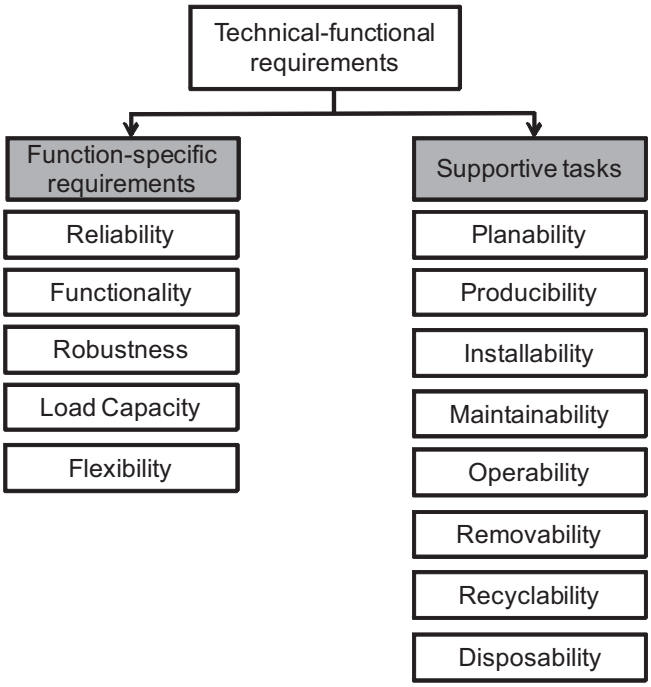

Fig. 4. Structuring in line with the dimension “technical functional aspects”

5 Advantages of the Model for Requirements Management

Requirements management is a crucial task to ensure customer-oriented product development. However, to cope with the variety of stakeholders and requirements a systematic and comprehensive requirements management is necessary. The system should assist in identifying all stakeholders and their requirements, structuring them adequately and preparing them for implementation by a QFD for example. For all this purposes the model presented can be applied. It allows surveying and structuring all requirements on the reference object and applying the QFD afterwards by structuring the requirements regarding their content and level of specification. For this, it is possible to consider, for instance, single groups of requirements which are single dimensions respectively categories of a dimension showing a same or similar level of specification.

The model also serves to check if all stakeholder requirements have been considered or if there are informational deficits. Dimensions or categories to which no requirements have been matched might give a hint that some stakeholders have not been considered.

6 Using the model within the Context of Quality Management

Within the framework of quality loop it has become clear that for customer orientation quality management has to include all phases of the life cycle [17]. Thus, gathering and structuring requirements is just the beginning. Nevertheless, this is crucial for the subsequent phases and has a systematic and integrative use of different methods of quality management. So, for reaching an optimization of its input as well as its output, one has to combine QFD and other methods (cf. Figure 5).

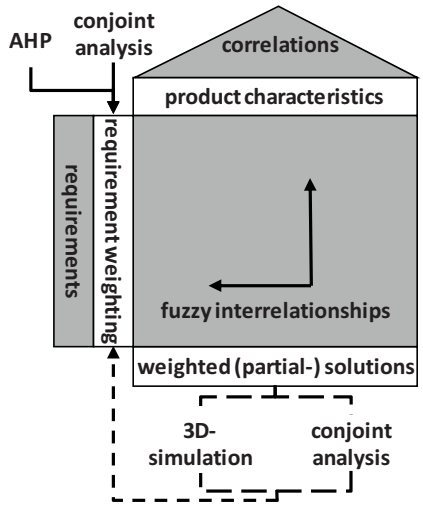

Fig. 5. Integrative application of QFD and other quality management methods

For receiving the weightings of the requirements for the first one the Analytic Hierarchy Process (AHP) and the conjoint analysis can be applied [18]. The conjoint analysis is the quantitative method which is mostly used for the development of (new) products [19]. One great advantage of this method is that it can rebuild the decisions respectively trades being made when making buying decisions and that it does not overburden the stakeholders [20].

Moreover, one can achieve an optimization of the planning and development process by receiving stakeholders' feedback to planned solutions instead of only requirements during the development process. Hence, the conjoint analysis might be useful as well. This allows connecting requirements and (possible) solution and by this focussing "the best" solution from the view of the stakeholders. In addition, this might contribute for stakeholders to being more precise in articulating requirements and also in even naming them. However, for this an IT-based support is necessary. Particularly, 3D-simulations offer the possibility for the stakeholders to see and consequently assess possible solutions respectively implementations of their requirements in early phases. Linking and advancing these methods is a major task for quality management in the future. Also in single branches some effort has been made in the last years, like for instance in the automotive industry. Nevertheless, the offered potential is not exploited yet.

Alternatively, the interfaces from QFD to the following stages in the product development have to be optimized. Bringing together the internal and external point of view is often difficult, i.e. the stakeholder requirements and their possible solutions with the construction department which is, among other things, responsible for defining tolerances. However, therefore stakeholder requirements have to be known and considered. Nevertheless, this stage of product development is often coined by the knowledge and expertise of the construction experts. Methods for optimizing the interfaces are necessary for linking these phases of product development and ensuring an adequate customer-orientation during the whole life cycle and consequently the quality loop. Receiving optimized products and processes by systematic requirements management and the following process control can only be allowed by a holistic consideration.

Acknowledgements

The authors wish to thank the Deutsche Forschungsgemeinschaft (DFG) for supporting their work within the framework of the Collaborative Research Centre 696.

References

1. van Beek, T.; Tomiyama, T.: Requirements for complex systems modeling, Proceedings of the 18th CIRP Design Conference, Enschede, the Netherlands (2008)
2. Klute, S.; Kolbe, C.; Refflinghaus, R.: Requirements Management a Premise for adequate Life Cycle Design. Glocalized Solutions for Sustainability in Manufacturing, Proceedings of the 18th CIRP International Conference on Life Cycle Engineering, Braunschweig, Germa ny, 02. 04.05.2011, Berlin, Springer, pp. 537 542 (2011)
3. Crostack, H. A.; Schlüter, N.; Mathis, J.; Jockisch, M.; Töllner, A.: Mit Struktur zur Be geisterung Kundenanforderungen an intralogistische Anlagen effektiv handhaben. In: Handling Das Magazin für Automation, Handhabungstechnik und Intralogistik, No. 6, pp. 96 98 (2008)
4. Janisch, M.: Das strategische Anspruchsgruppenmanagement vom Shareholder Value zum Stakeholder Value. Paul Haupt, Bern, Stuttgart (1993)
5. Klute, S.; Refflinghaus, R.: Structuring requirements as necessary premise for customer oriented development of complex products a generic approach. JIEM, Vol. 4, No. 3, pp. 523 537 (2011)
6. Pahl, G.; Beitz, W.: Engineering Design: A systematic Approach", Berlin, Springer Verlag, ISBN: 1846283183 (1996)
7. Ehrlenspiel, K.: Betriebsforderungen an Maschinen Bedeutung und Einteilung. Kon struktion. In: Zeitschrift für Produktentwicklung und Ingenieur Werkstoffe, Düsseldorf: Springer VDI Verlag, Vol. 29, S. 29 35 (1977)
8. Myers, J. H.; Shocker A. D.: The Nature of Product Related Attributes. In. International Journal of Research in Marketing, Vol. 5, pp 211 236 (1981)
9. Hentschel, B.: Die Messung wahrgenommener Dienstleistungsqualität mit SERVQUAL eine kritische Auseinandersetzung. In: Diskussionsbeiträge der Wirtschafts wissenschaftlichen Fakultät Ingolstadt, No. 3 (1990)
10. Kano, N.; Seraku, N.; Takahashi, F.; Tsuji, S.: Attractive quality and must be quality. In: Hinshitsu The Journal of the Japanese Society fo Quality Control, Vol. 14, No. 2, 1984, pp. 667 681 (1984)
11. Homburg, C.; Staritz, M.; Bingemer, S.: Commodity Differenzierung. Ein branchenüber greifender Ansatz, in: Commodity Marketing. Grundlagen Besonderheiten Erfahrungen. Hrsg. von M. und A. Geigenmüller, S. 33 54, Gabler, Wiesbaden (2011)
12. Hartel, D.: Consulting in Industrieunternehmen Praxisleitfaden mit Fallstudien. Olden bourg, München (2009)
13. Gautum, N.; Singh, N.: Lean product development: Maximizing the customer perceived value through design change (redesign). In: Journal of Production Economics, 114, pp 313 332 (2008)
14. Smith, D. J.: Reliability, maintainability and risk: practical methods for engineers, Oxford: Butterworth Heinemann (2005)
15. Blischke, W.; Murthy D.: Reliability: modeling, prediction, and optimization, Wiley (2000)
16. DIN 31051: Grundlagen der Instandhaltung. Juni (2003)
17. Pfeifer, T.; Schmitt,R.: Qualitätsmanagement: Strategien, Methoden, Techniken, Vol. 4, Hanser, München (2010)
18. Saaty, T.,: How to make a decision: The Analytic Hierarchy Process. In: European Journal of Operational Research, Vol. 48, No.1, pp. 9 26 (1990)
19. Pullmann, M. E., Moore, W. L., Wardell, D. G.: A comparison of quality function deploy ment and conjoint analysis is new product design. In: The Journal of Product Innovation Management, Vol. 19, No. 5, pp. 354 364 (2002)
20. Srinivasan, V.: A conjunctive compensatory approach to the self explication of multiat tributed preferences. In: Decision Sciences, Vol. 19, No. 2, pp. 295 305 (1988)

(Semi-)Automatic Categorization of Natural Language Requirements during Elicitation

Research Preview

Daniel Ott[1] and Eric Knauss[2]

[1] Research and Development, Daimler AG
P.O. Box 2360, 89013 Ulm, Germany
daniel.ott@daimler.com
[2] Department of Computer Science
University of Victoria
knauss@computer.org

Abstract. Context and motivation: Requirements of todays industry specifications need to be categorized for multiple reasons, including analysis of certain requirement types, like non-functional requirements, identification of dependencies among requirements or of relevant requirements. This is a pre-requisite for effective communication and prioritization of requirements in industry-size specifications. **Question/problem:** Because of the size and complexity of these specifications, categorization tasks must be done with the help of automatic mechanisms to minimize manual efforts. **Principal ideas/results:** In this work, we present a preview of our research to realize a framework, which automatically suggests fitting categories for a new requirement during the writing process of a new specification. This framework learns from the manually selected choices and will propose an even better selection of categories for the next requirement. It is the goal of our approach, to maximize the accuracy of the suggested categories and at the same time minimize the manual efforts of the author. **Contribution:** The underlying concepts of this framework and their interactions are described in detail in this work. In addition, we describe our plans to evaluate this framework in an experiment in cooperation with Mercedes-Benz.

Keywords: requirements, classification, categorization, natural language

1 Introduction

In current industry specifications it is essential to categorize requirements, partly because of their growing size and complexity, but for example according to Song and Hwong [7] also for a number of other reasons: Identification of requirements of different kinds (e.g. technical or non fun-functional requirements) is a necessity (1) for having specific guidelines for developing and analyzing these requirement types, (2) for architectural decisions, (3) for identifying equipment needed, its

quantity and permitted suppliers, and (4) for identifying dependencies among these requirements, especially to detect risks and for scheduling needs during the project. Related to this, Knauss et al. [4] report on the importance of classifying security-related requirements early in the project in order to prevent substantial security problems later. More recent, Ott [6] also reports the need to categorize requirements for inspection tasks to support reviewers with the detection of consistency or completeness defects over large document sets. Note that in this work, we use the term *classification* to refer to the algorithmic task of mapping requirements to topics and *categorization* to refer to more generally establishing a good mapping for a specific specification.

Efficient classification can enable focussed communication and prioritization of requirements. As the examples show, categorization of requirements allows filtering relevant requirements for a given important aspect. Considering large specifications, for example in the automotive domain (a single specification at Mercedes-Benz can consist of up to 50.000 requirements and headings [2]), it is necessary to minimize the manual efforts in categorization tasks.

Therefore, we present in this work a framework, which automatically suggests fitting categories for new requirements during the writing process of a new specification. In addition, our approach will learn and adjust its suggestions with each new requirement according to the user's choices. It is the goal of our approach, to maximize the accuracy of the suggested categories and on the same time minimize the manual efforts of the author.

Reading these ideas of a new approach, one could ask, "Why not writing the full document and then automatically classify all requirement at once, using a text classification algorithm like Naive Bayes?", like we have done in recent work with good results [6]. To answer this question, in the near future, we will conduct an experiment in cooperation with Mercedes-Benz. In the experiment, Mercedes-Benz developers will write and categorize functional requirements for different car systems. By observing their specific actions, we will investigate and compare the performance and manual efforts of the participants for categorizing requirements depending on the level of support: a) fully manual, b) fully automatic, or c) semi-automatic, as aimed for in our framework.

In Section 2 we describe related work, before we present our framework (the planned user interactions and classification mechanisms) in Section 3. Finally, we share the details of our planed experiment with Mercedes-Benz in Section 4.

2 Related Works

In this section, we discuss a spectrum of approaches for classification of requirements. On the one side of this spectrum are approaches that are based on purely manual classification, as supported by most state-of-the-art requirements management tools. Analysts specify the classification of requirements in a user-defined attribute. As one such example, Song and Hwong [7] report about their experiences with manual categorizations of requirements in a contract-based sys-

tem integration project. The contract for this project contains over 4,000 clauses, which are mostly contract requirements.

On the other side of the spectrum are approaches that classify requirements only based on automatic classification. Examples include QuARS tool by Gnesi et al. [1], which automatically detects linguistic defects like ambiguities, using an initial parsing of the requirements. Thereby, QuARS creates a categorization of requirements to topics as a byproduct.

Especially when based on machine learning, such approaches face the problem to obtain large enough training sets in sufficient quality. Knauss et al. [4] evaluate to what extent security-relevant requirements can be automatically identified in specifications based on Naive Bayesian Classifiers. Accordingly, satisfactory results can be achieved, if both training and testing data was derived from the same specification. This is probably due to the fact that writing style and domain specific concepts have a strong impact on the classifier's performance. Ott [6] reports similar results for automatic classification of requirements in multiple categories for supporting review activities. For this reason, Ko et al. [5] propose to automatically create the training data for topic classification. Based on a clustering algorithm they categorize requirements and use these to train Naive Bayesian classifiers. The evaluation results of this approach are promising, but only based on small English and Korean specifications (less than 200 sentences).

Hussain et al. [3] developed the tool LASR that offers an interactive modus for supporting groups in annotation tasks. By not relying on a fully automatic classification approach they mitigate the problem of insufficient training data. In contrast to our work, they try to support a group in collaboratively creating and agreeing on a categorization, whereas we focus on supporting single annotators with a special focus on cost and quality, as well as continuous improvement.

3 Framework for Learning

We define a topic as any crosscutting concern that demands for the ability to filter related requirements. Examples include qualitative requirements, such as performance or security-relevance, crosscutting design issues or constraints such as temperature, or regulatory concerns. A requirement can be assigned to a set of topics. Technically, this can be done by adding an attribute *topic* to the requirement and specify relevant topics as a comma separated list.

When requirements are categorized into topics, certain tasks (e.g. creating a security concept, reviewing, priotizing) become much simpler. As shown by related work, automatic topic classification of natural language requirements is technically feasible but prone to writing style and domain specificity. The main reason for these problems is the lack of sufficient training data in high quality. For this reason, the integration of such algorithms in the requirements specification process needs to be considered carefully.

To get a good categorization, our framework needs to support four main use cases: The framework should support the author of a requirements document in choosing topics during the documentation of requirements, it should propose

relevant topics when the user chooses a topic for a given requirement, it should allow the user to add new topics to the framework, and it should support assigning topics to a set of requirements that are already documented.

A framework for requirements categorization needs to be able to learn, because otherwise it could not adjust to domain specific concepts or writing style. This learning can be observed on several levels. First, users learn a suitable system of topics during working with the requirements. Second, the classification framework itself should learn from previous classifications and gain more and more accuracy in proposing relevant topics. Finally, the framework needs to support learning new topics, i.e. let the user add new topics without breaking the quality of proposing existing topics.

The value of requirements categorization depends on its quality. For example, consider designing a security concept. In this case it is very important that all security relevant requirements are identified. Moreover, the value of the topic classification needs to be higher than the cost to create it.

Generally, a framework can offer different modi of operation. First of all, it could define a process and instructions for manual classification. We assume that this modus can generate a high quality categorization at high cost. Secondly, it could completely rely on automatic classification. By eliminating the need for human intervention, the cost to create the classification is minimal. Based on the lack of high quality training data, the quality of the topic classification in this modus might just be too low for many tasks. Thus, we are especially interested in a third modus, where the system recommends relevant topics and allows the user to interact by confirming or rejecting recommendations. This interaction can be used to train supervised learning algorithms such as Bayesian classifiers.

Figure 1 shows details to the individual processing steps of our proposed realization of this framework. The chosen pre-processing, post-processing and classification steps have many alternatives, but after a comparison, we got the best results in previous work [6] with the illustrated setting for German natural-language specifications from Mercedes-Benz. We reuse this setting in the current work, since the used specifications in the experiment (see Section 4) in the near future have the same characteristics. We assume that a requirement can be classified to multiple topics. Therefore, we train a binary classifier for each topic, which decides if a requirement is relevant or not for a certain topic.

For k-gram indexing, each word of each requirement is separated in each ongoing combination of k letters and the classifier is then trained with these k-grams instead of the whole words. Based on external training data (manually annotated requirements), the svm then calculates for each topic a so

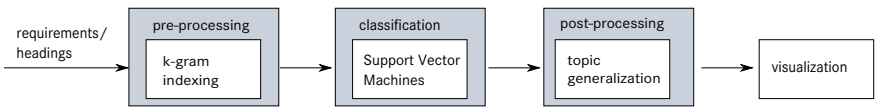

Fig. 1: Processing Steps

called maximum-margin hyperplane with the greatest separation between the training requirements belonging to the topic and the ones that do not. With this hyperplane the svm can assign a new requirement to the topic or not. Finally, the post-processing step called "topic generalization" takes the structure of Mercedes-Benz specifications into account. All specifications at Mercedes-Benz are written using a template, which provides a generic structure and general requirements, and are later filled with system specific contents. Because of this structure, we assume that if a heading was assigned to a topic, then we can also assign each of the requirements and subheadings below it to this topic. This allows to relate requirements represented by tables or figures to the topics of their headings. Please refer to [6] for a more detailed description of these steps.

4 Planed Evaluation

The purpose of this experiment is to test in which way automatic classification helps to reduce effort and to increase quality of topic classifications in industrial requirements specifications. We are also interested in learning effects. For this reason, we plan to relate initial training and user/framework interaction to learning, i.e. changes of effort and quality over time.

Consequently, we define the following independent variables: *Automatic classification support* can be activated or deactivated. *Initial training* of the framework might provide better suggestions in the beginning, but take longer to adjust to a special problem domain. *User confirms classification* determines if the user confirms the classification and potentially overrides automatic classifications.

We will monitor the following dependent variables: *effort* to provide a requirements specification with topic classification and *quality* of the topic classification. Finally, we control for the quality of the requirements specification itself.

The independent and dependent variables lead to the following *hypotheses*:

H1: Automatic classification leads to higher quality than manual classification.
H2: Automatic classification leads to less effort than manual classification.
H3: Starting with an initially trained classifier leads to better classifications than starting with an untrained automatic classifier.
H4: An untrained classifier adjusts faster to the problem domain than an initially trained classifier.
H5: The combination of automatic classification and user confirmation leads to higher quality of classifications than automatic classification.
H6: The combination of automatic classification and user confirmation leads to less effort than manual classification.
H7: Users profit from the framework's feedback by learning categorization.

We plan to evaluate our hypotheses in an experiment and semi-structured follow-up interviews with the participants (especially for covering hypothesis 7 and for determining a good ratio of $\frac{\text{effort}}{\text{size}}$ for practical use). For this experiment, we assign practitioners to four groups and provide all participants with an exemplary implementation of our framework (Table 1). Participants will interact in

Table 1: Experiment design: Groups of participants and independent variables.

Group	*Automatic classification support*	*Initial training*	*User confirms classification*
1	no	no	yes
2	yes	yes	no
3	yes	no	yes
4	yes	yes	yes

four different ways with our framework to categorize requirements while writing a requirements specification. Group 1 writes and classifies all requirements parallel without any tool support. Group 2 focuses on writing the requirements and completely relies on the framework for classification. Group 3 writes the requirements and classifies them interactively with the framework. In this case, the framework is not initialized with any training data and needs to learn the classification from the user. Group 4 works similar to group 3 but in this case the framework is initialized with training data from old requirements specifications. Both, in group 3 and 4, the framework learns through interaction with the user.

We establish a ground truth based on expert classification and measure the group's classification quality by comparing against it. To ensure sufficient quality of the ground truth, we let two experts classify the requirements iteratively and measure their agreement in their classification. If the agreement level is below a threshold (based on inter-rater agreement, e.g. Cohen's kappa), the raters need to discuss situations where they disagree and improve for the next iteration.

References

1. S. Gnesi, G. Lami, G. Trentanni, F. Fabbrini, and M. Fusani. An automatic tool for the analysis of natural language requirements. *International Journal of Computer Systems Science & Engineering*, 20(1):53–62, 2005.
2. F. Houdek. Challenges in Automotive Requirements Engineering. *Industrial Presentations at REFSQ 2010, Essen*.
3. I. Hussain, O. Ormandjieva, and L. Kosseim. Lasr: A tool for large scale annotation of software requirements. In *Empirical Requirements Engineering (EmpiRE), 2012 IEEE Second International Workshop on*, pages 57–60. IEEE, 2012.
4. E. Knauss, S. Houmb, K. Schneider, S. Islam, and J. Jürjens. Supporting requirements engineers in recognising security issues. *Requirements Engineering: Foundation for Software Quality*, pages 4–18, 2011.
5. Y. Ko, S. Park, J. Seo, and S. Choi. Using classification techniques for informal requirements in the requirements analysis-supporting system. *Information and Software Technology*, 49:1128–1140, 2007.
6. Daniel Ott. Automatic requirement categorization of large natural language specifications at mercedes-benz for review improvements. In *Requirements Engineering: Foundation for Software Quality*, 2013.
7. X. Song and B. Hwong. Categorizing requirements for a contract-based system integration project. In *Requirements Engineering Conference (RE)*, pages 279–284. IEEE, 2012.

Guiding Requirements Elicitation using a Prioritization Framework

Norman Riegel[1]

[1] [1]Fraunhofer IESE, Fraunhofer Platz 1, 67663 Kaiserslautern, Germany
norman.riegel@iese.fraunhofer.de

Abstract. In the area of information systems (IS), one major goal of many IS development projects is to support the business processes of an enterprise in order to optimize business performance. In this area, business-process-driven requirements engineering (BPRE) plays an important role, combining business process management activities and requirements engineering (RE) activities. Thereby, decisions have to be made in order to decide which business processes and derived requirements are the most valuable ones and therefore, where effort should reasonably be spent for elicitation and analysis activities. In order to focus on those requirements that promise the greatest value regarding certain criteria, typically prioritization techniques are used. However, most prioritization approaches are too generic to provide an appropriate solution for the BPRE context. Furthermore, they do not support the requirements engineer during elicitation in order to depict those requirements valuable for further analysis and refinement. To tackle this problem, the idea of applying different models during prioritization is applied. This is expected to reduce unnecessary (RE) activities by focusing on the most important requirements. In this paper, we introduce a prioritization method from our BPRE prioritization framework to cope with this problem.

Keywords: Requirements Prioritization, Requirements Elicitation, Business-Process-Driven Requirements Engineering, Information Systems

1 Introduction

In the area of information systems (IS), one important goal of many IS development projects is to support the business processes of an enterprise in order to optimize business performance. In these types of projects, business-process-driven requirements engineering (BPRE) plays a major role, combining business process management activities (e.g., business process design, optimization, reengineering) and requirements engineering activities (cf., e.g., [1], [2]). Thereby, the analysis and specification of business processes is a major task in order to derive fine grained software requirements from them. The requirements elicitation process therefore typically starts on the level of business processes and during further progress more detailed requirements on different levels of abstraction [2] (e.g., business activity descriptions, detailed system functions) are derived. Due to the large number of business processes (and derived requirements) even existent in small and medium-

sized enterprises it is inevitable to focus the requirements elicitation process. Elicitation must therefore be designed in the most efficient way, i.e. eliciting the requirements in an optimal order as much as possible. Ideally, effort should be put on the most promising requirements in terms of business value while at the same time minimizing elicitation effort for less important requirements. Thereby, decisions have to be made in order to decide which business processes and derived requirements are the most valuable ones and therefore, where effort should reasonably be spent for elicitation and analysis activities. Typically, prioritization techniques are used to focus on those requirements that promise the greatest value regarding certain criteria (e.g., benefits and costs). In the literature, numerous prioritization approaches can be found, differing for example in terms of their procedure, complexity and calculations (cf., e.g., [3]). However, the application of these techniques in practice is problematic in the depicted BPRE context. Most prioritization approaches are too generic to provide an appropriate solution for the BPRE context [4]. For example, the specifities of the different requirements types are not regarded and assessment criteria for the different requirements types are missing, making an appropriate assessment hardly possible. Also, most approaches are intended to be applied after the elicitation process has finished, instead of supporting prioritization already during elicitation. This has the consequence that effort is wasted on (RE) activities of minor importance, e.g., on eliciting and analyzing requirements of minor importance in interviews and workshops [4]. To cope with this problem, we developed a prioritization framework (Figure 2) based on requirements derived from the BPRE context [5]. The framework is intended to support requirements engineers in overcoming this problem by applying different models during prioritization. This idea is expected to reduce unnecessary (RE) activities by focusing on the most important requirements during elicitation already.

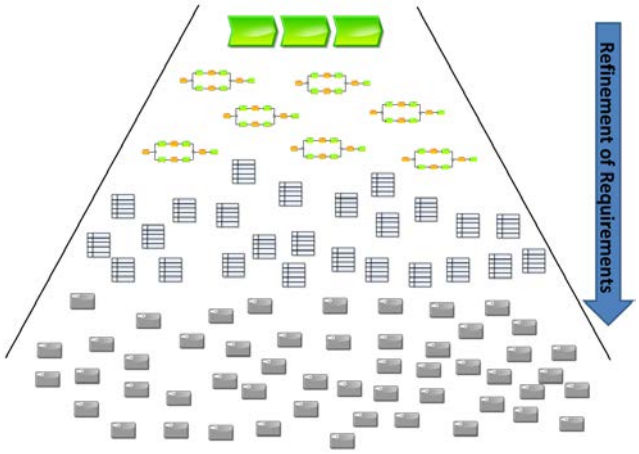

Fig. 1. Requirements hierarchy.

In this paper we want to focus on the usage of the framework. The remainder of this paper is structured as follows: Section 2 briefly describes related work in this area,

section 3 depicts the framework itself and describes its application in BPRE, and section 4 finally gives an outlook on our future work in this area.

2 Related Work

Many prioritization approaches have already been proposed in the literature (cf., e.g., [3]), which differ quite significantly in terms of complexity, procedures, and calculations. Often cited ones include for example the Analytic Hierarchy Process (AHP) [7] and the Cost-value Approach [8], which is based on the preceding, Hierarchical Cumulative Voting (HCV) [9], the Kano Model [10], Quality Function Deployment (QFD) [11], Quantitative WinWin [12] and Wieger's Method [13]. To assess in detail the applicability and the shortcomings of existing prioritization approaches in BPRE, we evaluated about 30 prioritization approaches (including the ones mentioned before) from the literature against a set of basic requirements that an approach should fulfill in order to be applied in BPRE (see our survey in [4]). These requirements were derived from typical characteristics and challenges we found in related literature and experienced in our BPRE industry projects in the past. To summarize the results described in [4], none of the regarded approaches were able to fulfill all requirements completely. It turned out that most prioritization techniques are too generic to provide an appropriate solution for the BPRE context. This means, that for example the specifities of the different requirements types are not regarded, specific assessment criteria are not provided and information about stakeholders to be involved is missing. Also, it became apparent that most approaches are intended to be applied after the elicitation process has finished, instead of supporting prioritization already during elicitation. This means, that most approaches take as input for the prioritization an already specified (final) set of requirements instead of using prioritization to focus elicitation effort. Based on the results of this survey, the BPRE prioritization framework was developed [5].

3 Application of the BPRE Prioritization Framework

The BPRE prioritization framework [5] (Figure 2) is a set of utilities that enables requirements engineers to appropriately prioritize requirements that are elicited during BPRE on different abstraction levels (i.e., requirements emerging in the hierarchy of business processes). This means that during elicitation, the framework is applied to prioritize the requirements on each abstraction level in order to determine the most valuable requirements to be refined further. The framework consists of three models – the *Prioritization Object Model (POM)* (the *POM* is based on our work in [14]), the *Value Model (VM)*, and the *Stakeholder Model (SM)* – and also encompasses a tool-supported *Prioritization Method*.

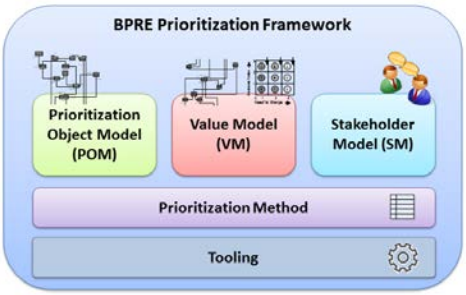

Fig. 2. Building blocks in the prioritization framework for BPRE (based on [5]).

The models are reference models and formalize best practice information for prioritization. Thus, this information does not have to be created anew for each prioritization setting. The three models are the basis and input for the *Prioritization Method*. Figure 3 depicts the interplay between the elements of the different components of the framework in a meta-model.

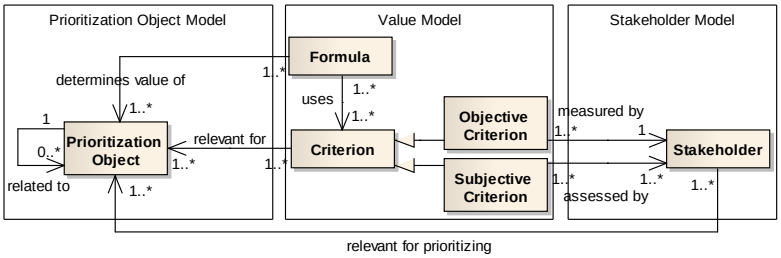

Fig. 3. Interrelationships between the elements of the framework.

3.1 General application of the framework

Figure 4 depicts an overview of the general application of the framework. In the *Preparation Phase* a first tailoring of the different models (*POM*, *VM* and *SM*) takes place based on the project or general goals of the business (e.g., determination of relevant requirements types and criteria and mapping of stakeholders). In the *Requirements Elicitation Phase* these tailored models are used to conduct a prioritization-aware elicitation of requirements, i.e. elicitation and prioritization are performed alternately. This means that after each elicitation step (meaning the elicitation of requirements on a certain level of abstraction as depicted in Figure 1), prioritization is used in order to determine which of the elicited requirements will be analyzed and refined in the next elicitation step.

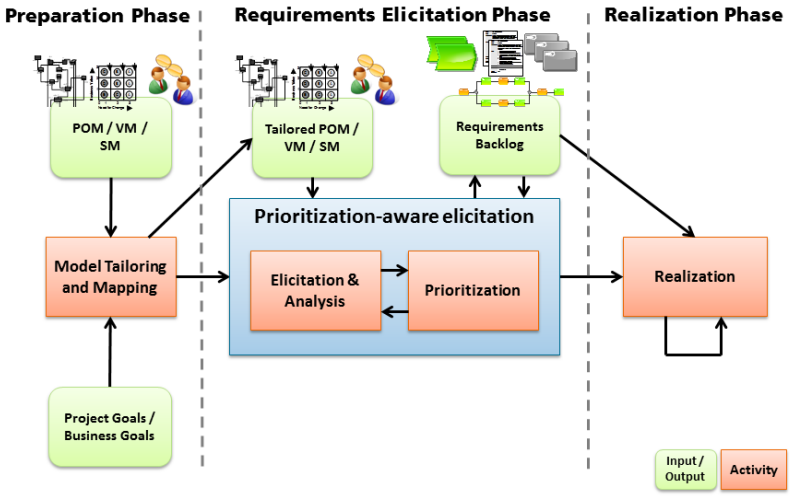

Fig. 4. Overview of the application of the prioritization framework for BPRE.

Thereby, the elicited and prioritized requirements are stored in a requirements backlog. The requirements backlog therefore consists of requirements on different levels of abstraction, whereas the requirements on the lowest abstraction level will serve as input for the *Realization Phase*. The *Realization Phase* itself describes the activities in subsequent steps in the software development process, e.g. design and implementation activities, which will not be further elaborated here.

In order to illustrate the interplay between elicitation and prioritization in the *Requirements Elicitation Phase* better, Figure 5 shows an example for the two requirements types *business process* and *business activity*: (I) During elicitation activities, business processes are identified as black boxes at first. (II) These business processes are then prioritized and stored in the process backlog. (III) The high priority business processes are then analyzed in detail and (black box) business activities are successively derived from them. (IV) These business activities are then prioritized and stored in the activity backlog. (V) The high priority business activities are then analyzed in detail and (black box) system functions are successively derived from them. Thus, some elicitation effort is needed in order to do the prioritization. Typically this includes at least the identification of requirements without specifying them in more detail. For example, a business process is prioritized as black box without knowing the details of its workflow or a business activity is prioritized as black box without knowing the details of its sequence flow.

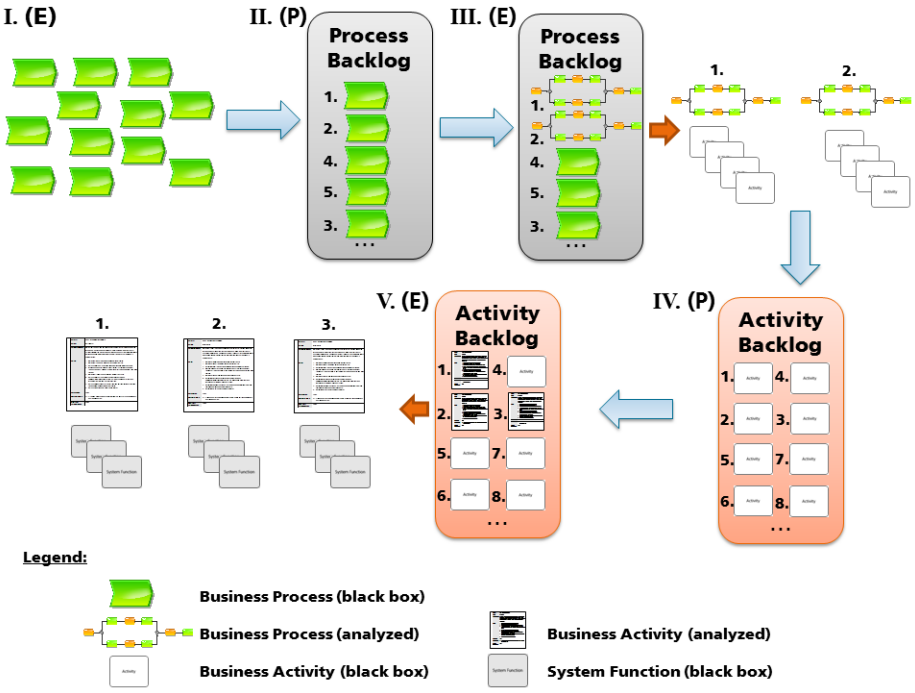

Fig. 5. Interplay between Elicitation (E) and Prioritization (P).

3.2 The Prioritization Method

In the following, we will focus on the usage of the BPRE prioritization framework in the *Requirements Elicitation Phase* and omit the details of the *Preparation Phase* in this paper. Therefore we describe the application of the *Prioritization Method* of the framework during requirements elicitation in a BPRE setting. We will also provide an example[1], which makes use of content already provided by the different parts of the framework (see extracted content in Figure 6). As we abstract from the tailoring activities in the *Preparation Phase* in this paper, the prerequisite for the method is that on the basis of the *POM*, it has already been decided which requirements types are going to be elicited during the BPRE project. For this, the *POM* comprises requirements types (i.e., the prioritization objects) and their relationships typically addressed in the IS domain. Then, the prioritization method as described below is applied on each abstraction level of requirements.

[1] The example was constructed out of a real case study where we applied parts of our method. However, as the method was not exactly applied in the case study as described here, we omit further details about this study in order to not confuse the reader.

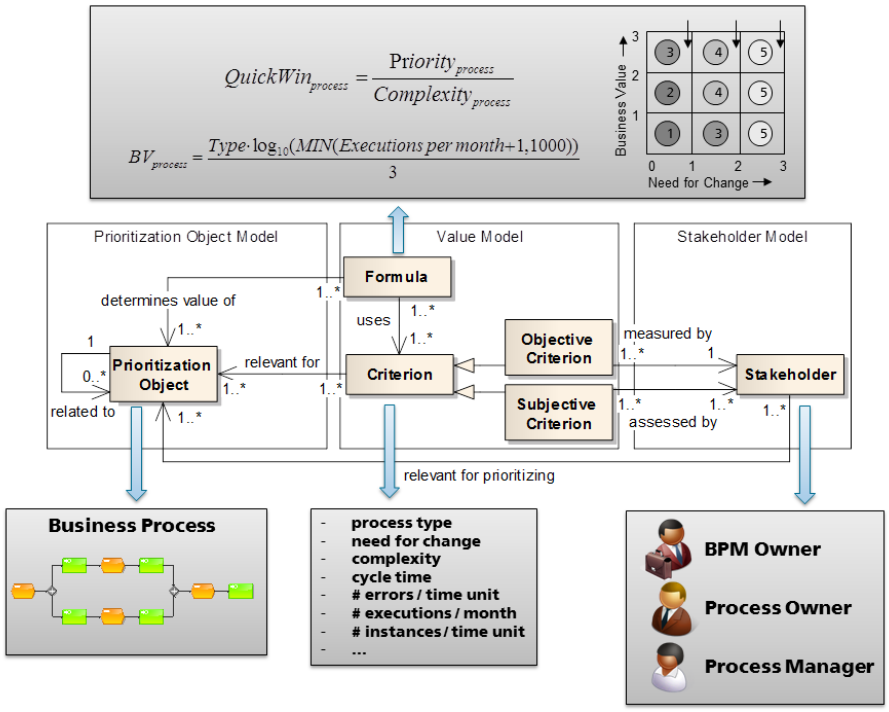

Fig. 6. Exemplary prioritization information extracted from the different models.

Step 1: Determination of requirements types

Input: Requirements elicited on current level of abstraction

Description: The purpose of this step is to determine the requirements type which is going to be prioritized at the current level of abstraction of elicitation. This is necessary in order to determine the related assessment criteria and relevant stakeholders during the next steps of the method. The hierarchy of the requirements types is defined in the *POM* as described above.

Output: Requirements type to be prioritized

Example: As an example, suppose we are on the business requirements level during elicitation for an IS development project. The business processes are only identified as black boxes here. The object *business process* is an element in the *POM* and the current object for prioritization (Figure 6). At this early stage of the elicitation phase, the purpose of prioritization is to filter the business processes that should be further considered and analyzed in detail. Figure 7 shows an example of such a list of business processes.

Position	Business Process
1	Bill payment
2	Application for business trip
3	Effort report
4	Application for compensatory time
5	Expense account

6	Billing of hospitality
7	Procurement
…	…

Fig. 7. Identified business processes at an early stage of elicitation.

Step 2: Determination of assessment criteria

Input: Requirements type to be prioritized

Description: The purpose of this step is to identify the assessment criteria that can be used to appropriately rate the requirements types identified in Step 1. These criteria are located in the *VM* and linked to the relevant prioritization objects in the *POM*, as shown in Figure 3. Criteria may be objectively measured or subjectively assessed by the stakeholders involved in the prioritization.

Output: Objective and subjective assessment criteria relevant for the requirements type of interest

Example: In our example, criteria that can be found in the *VM* for prioritizing business processes are, for example: (1) the *type* of process, (2) the *number of executions* per month, (3) the *need for change* and (4) the *complexity* of the process (Figure 6). (1) and (2) can be determined objectively, whereas (3) and (4) must be assessed subjectively by the stakeholders in this example. Information about these different criteria can be found in [1] and will not be provided in detail here.

Step 3: Determination of stakeholders

Input: Requirements type to be prioritized

Description: The purpose of this step is to identify the stakeholders who are needed to prioritize the determined requirements types according to the selected criteria. This should ensure that only stakeholders in a reasonable mixture rate the requirements as well as that even black box requirements are rated as reasonable as possible. Typical roles and stakeholders in BPRE are expressed in the *SM*. The elements of the *SM* are linked to the elements in the *POM*; thus, it can be determined which stakeholders are required for doing the prioritization.

Output: Stakeholders relevant for assessing the requirements type of interest

Example: In our example, suppose the relevant roles provided by the *SM* for prioritizing business processes are the *BPM owner* (i.e., the role responsible for the whole BPM project), the *process owners* (i.e., the role responsible for a process), and the *process managers* (i.e., the role responsible for the control of a process) (Figure 6). Of course, these roles have to be mapped to corresponding roles / persons in the respective organization beforehand in the *Preparation Phase*.

Step 4: Determination of the prioritization formula

Input: Requirements type to be prioritized

Description: To determine the ranking order of the requirements, a calculation formula is required, which will be determined in this step. For this, the *VM* provides calculation methods for combining the objective and subjective criteria. The calculation formulas are dependent on the prioritization objects because the number and type of the criteria vary for the different requirements types. Thus, the relevant formula for the actual elicitation stage has to be determined. Of course, the priorities of the more abstract requirements might influence the priorities of the more fine-grained ones. This is also represented in the calculation formulas.

Output: Prioritization formula for determining the priority value of the requirements type

Example: In our example, suppose the formula for calculating the ranking order of business processes provided as part of the *VM* is the *QuickWin* value (see Figure 6), which is calculated by:

$$QuickWin_{process} = \frac{Priority_{process}}{Complexity_{process}}$$

Thus, we have to calculate the *priority* value and juxtapose it to the *complexity* value for each process to get the *QuickWin* values. The *priority* of a business process is calculated by a combination of the two dimensions *business value* and *need for change* (see Figure 8). This is also provided by the *VM*. Information about this calculation can be found in [1] and will not be provided in detail here.

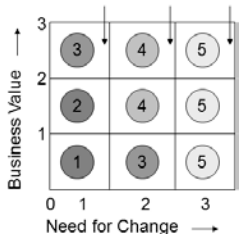

Fig. 8. Prioritization matrix (based on [1]).

Step 5: Performance of the actual prioritization

Input: Requirements and relevant assessment criteria, stakeholders and prioritization formula(s)

Description: Having identified all relevant information from the models that is needed for prioritizing the requirements, the purpose of this step is the actual prioritization, i.e., the assessment and calculation of the ranking order. For this purpose, objective criteria are measured and subjective criteria are rated by the determined stakeholders, with the values being rated on different scales, depending on the type of criterion. The assessment scheme is also a part of the *VM*. After the assessment, the ranking order is calculated according to the formulas determined before.

Output: Ranked order of prioritized requirements

Example: In our example, the *type* of all identified business processes must be determined, as well as the *number of executions* per month *(#Exec/m)*. Furthermore, the identified stakeholders, i.e., the *BPM owner*, the *process owners*, and the *process managers*, are asked to rate the *need for change (NfC)* for each process from their point of view, as well as its *complexity (Compl)*. After that, the *QuickWin* value for each process is calculated according to the formulas determined from the *VM*. Finally, a ranked order of the business processes is the result (see Figure 9).

Business process	Type	#Exec/m	BV	NfC	Prio	Compl	QuickWin
Bill payment	2	1300	2	2	4	1	4,00
Appl. for business trip	3	500	3	3	5	2	2,50
Effort report	2	200	2	1	2	1	2,00
Appl. for comp. time	2	200	2	1	2	1	2,00

Expense account	1	600	1	3	5	4	1,25
Billing of hospitality	1	900	1	1	1	1	1,00
Procurement	3	1100	3	3	5	5	1,00
…							

Fig. 9. Ranked order of prioritized business processes.

Step 6: Decision on how to proceed next

Input: Ranked order of prioritized requirements

Description: After the ranking order has been calculated, the purpose of this step is to decide how to proceed next. As prioritization goes hand in hand with elicitation, it has to be decided which of the prioritized requirements will now be refined further to derive more fine-grained requirements. Furthermore, decisions about going one step back in the requirements hierarchy also have to be made. This means that if it is reasonable, new iterations of the top-down elicitation can be started based on the priorities calculated on the previous level of hierarchy (see Figure 10).

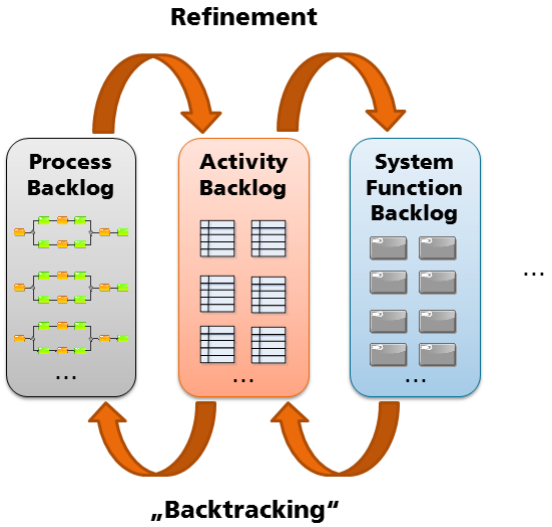

Fig. 10. Refinement and “backtracking”.

Output: Decision, on which requirements to refine further or if going back in the hierarchy is more valuable

Example: In our example the most valuable business process to analyze and refine would be the bill payment process because of the highest *QuickWin* value. Also it could be decided to analyze more than one process in parallel. This decision could be based on budget and time constraints or available resources. The analysis would lead to a set of more fine grained requirements which will be prioritized again with the same procedure.

4 Conclusion and Future Work

In this paper, we described the application of the *Prioritization Method* of our BPRE prioritization framework which was introduced in [5]. The purpose of this method is to reduce unnecessary (RE) activities by focusing on the most important requirements by applying different models during prioritization. It assists the requirements engineer during iterative elicitation in deciding, which requirements are most valuable to be analyzed and refined further. There are still a couple of open research issues concerning the different elements of the framework.

First of all, we have to finalize and evaluate our reference models because they form the underlying foundation of the whole approach. Especially for the *VM*, we have to identify further criteria via literature research and expert surveys, iteratively enhancing the model in order to find relevant criteria for all requirements types defined by the *POM*. Concerning the *SM*, we plan to investigate in several controlled experiments which roles are really relevant for assessing certain requirement types, respectively what the differences in the prioritization result are if stakeholders in different mixtures prioritize the requirements in order to determine the optimal composition.

Second, the calculations in the *VM* have to be extended. Here, already existing approaches in the literature (e.g., approaches as assessed in [4]) could be adapted or combined to allow taking into account the specificities of the different (objective and subjective) value criteria of the *VM*.

Third, we have to formalize the *Prioritization Method* described in this paper in order to provide precise or even automatable guidance. This should be aligned with the development of a supporting prioritization tool.

Last but not least, validation of the application of the approach in BPRE is planned in controlled experiments and industrial case studies with regard to requirements engineering efficiency and prioritization effectiveness.

References

1. Riegel, N., Adam, S., Uenalan, O.: Integrating Prioritization into Business Process-driven Requirements Engineering. In: REFSQ ‘10 Workshop Proceedings, pp. 113--118 (2010)
2. De La Vara, J. S., Díaz, J. S.: Business process-driven requirements engineering: a goal-based approach. In: CAiSE 2007 Workshop Proceedings, Tapir Academic Press (2007)
3. Daneva, M., Herrmann, A.: Requirements Prioritization Based on Benefit and Cost Prediction: An Agenda for Future Research. In: Proc. of RE 2009, pp. 125--134 (2009)
4. Riegel, N., Doerr, J., Hummel, O.: Tackling Prioritization in Business-Process-Driven Software Development. In: REFSQ ‘12 Workshop Proceedings, pp. 175--180 (2012)
5. Riegel, N.: Model-Based Prioritization in Business-Process-Driven Software Development. In: Proc. of Doctoral Symposium RE 2012 (2012)
7. Saaty, T. L.: The Analytic Hierarchy Process. McGraw-Hill, New York (1980)
8. Karlsson J., Ryan, K.: A Cost-Value Approach for Prioritizing Requirements. In: IEEE Software 14, vol. 5, pp. 67-74. (1997)
9. Berander, P., Jönsson, P.: Hierarchical Cumulative Voting (HCV) - Prioritization of Requirements in Hierarchies. International Journal of Software Engineering and Knowledge Engineering 16, vol. 6, pp. 819--849 (2006)

10.Berger, C. et al.: Kano's Methods for Understanding Customer defined Quality. Center for Quality of Management Journal 4, vol. 2, pp. 3--36 (1993)
11.Akao, Y.: Quality Function Deployment: Integrating Customer Requirements into Product Design. Productivity Press (1988)
12.Ruhe, G., Eberlein, A., Pfahl, D.: Trade-off analysis for requirements selection. In: Int. Journal of Software Engineering and Knowledge Eng., vol. 13, no.4, pp. 345-366. (2003)
13.Wiegers, K. E.: First Things First: Prioritizing Requirements. Software Development 7, vol. 9, pp. 48--53 (1999)
14.Adam, S., Riegel, N., Gross, A., Uenalan, O., Darting, S.: A Conceptual Foundation of Requirements Engineering for Business Information Systems. In: Proc. of BPMDS 2012 (2012)

Part II

REFSQ 2013 Empirical Track Proceedings

5 Preface

Editors

Daniel Berry
University of Waterloo, Canada, dberry@uwaterloo.ca

Norbert Seyff
University of Zurich, Switzerland, seyff@ifi.uzh.ch

Empirical Track

Daniel Berry[1], Norbert Seyff[2]

[1]University of Waterloo, Canada, dberry@uwaterloo.ca
[2]University of Zurich, Switzerland, seyff@ifi.uzh.ch

Following previous years' successes, we were given the opportunity to repeat the Empirical Track at REFSQ 2013 and issued a call for the following kinds of submissions:

- Alive Empirical Study: a controlled experiment, requiring no more than 90 minutes, that involves all REFSQ participants who want to participate,
- Online Questionnaire: an online questionnaire (survey), designed to require no more than 30 minutes, that is promoted at REFSQ and that can be filled out by all interested REFSQ participants, in their spare time during the conference, and
- Empirical Research Fair Proposal: a description of an empirical study that a researcher would like to conduct in an industrial setting or vice versa.

Overall we received eleven high quality submissions, of which we selected eight to be presented during the Empirical Track: one Alive Empirical Study, four Online Questionnaires and three posters in the Empirical Fair. In addition, the researchers conducting the Alive Empirical Study and one Online Questionnaire exhibited posters at the Empirical Fair.

1 Alive Empirical Study

REFSQ 2013 again issued a call that offers an opportunity to conduct an empirical study during the conference itself. The goals of this opportunity, besides that of permitting to conduct the experiment, are to raise awareness for the necessity and benefits of empirical studies and to show that participating in them is not dangerous to one's health. Furthermore, we wanted to bring together the community of researchers and practitioners who are interested in empirical studies. Therefore, we selected the experiment titled *A Quasi-Experiment for Studying the Effect of Experience on Elicitation Effectiveness*, organized by Alejandrina M. Aranda, Oscar Dieste, and Natalia Juristo to be conducted at REFSQ 2013. The experiment evaluated the effect of experience on requirements elicitation. The authors played the role of the customer, and each volunteering participant played the role of an analyst.

2 Online Questionnaires:

At REFSQ an online questionnaire (survey) is supposed to be designed to be filled out by all interested REFSQ participants, in their spare time at the conference, during breaks, etc. It should require no more than 30 minutes in order to participate. The following online questionnaires were selected for REFSQ 2013:

- *Using a Pattern Catalogue in Requirements Engineering Activities* by Cristina Palomares, Xavier Franch, and Carme Quer
- *A Survey on Lessons Learnt in Requirements Engineering* by Ibtehal Noorwali and Nazim H. Madhavji
- *Why Do You Install Apps?* by Mariano Ceccato, Alessandro Marchetto, Anna Perini, and Angelo Susi
- *Prioritization of Security Requirements for Cloud Computing* by Georg Herzwurm, Norman Pelzl, Benedikt Krams, and Sixten Schockert

3 Empirical Research Fair:

It is clearly understood in the RE community that case studies of industry projects are critical for in-depth understanding of both: (a) the phenomena occurring in projects, processes, systems, and services and (b) the impact of RE methods on the quality, cost, and deliverability of systems. Therefore, in the Empirical Fair, practitioners were asked to propose studies that their organizations would like to have conducted, and researchers were asked to propose studies that they would like to conduct in industry. The Empirical Fair was a meeting point to match the demand and supply of empirical studies among researchers and practitioners. To encourage industry participation, the format of this session was a match-making session in which the authors of the accepted proposals present posters on their intended case studies and the audience viewed them and entered a good discussion on the studies goals, benefits, and procedure. The following five proposals were presented with the help of posters during the fair:

- *Automation Supported Requirements Prioritization for Software Testing Purposes* by Michael Felderer, Boban Celebic, Christian Haisjackl, and Ruth Breu
- *Understanding Technical Debt for Better Requirements Prioritization* by Maya Daneva
- *Necessity of Electronic Requirements Negotiation* by Georg Herzwurm, Mareike Schoop, Benedikt Krams, and Sixten Schockert
- *Using a Pattern Catalogue in Requirements Engineering Activities* by Cristina Palomares, Xavier Franch, and Carme Quer
- *A Quasi-Experiment for Studying the Effect of Experience on Elicitation Effectiveness* by Alejandrina M. Aranda, Oscar Dieste, and Natalia Juristo

Acknowledgements

We would like to thank Jörg Dörr and Andreas L. Opdahl, the Program Committee Co-Chairs of REFSQ 2013, for giving us the opportunity to perform the Empirical Track. We sincerely thank all the authors of the empirical track for their contributions and for their hard work in preparing, conducting, and analyzing the empirical studies. Everybody that is involved in empirical studies knows that it is a lot of work. Furthermore, we would like to thank the members of the program committee of the Empirical Track for their valuable reviews that made it possible to select high quality contributions. Finally, no empirical study can be successful without active participations of people in the study. We would like to express our sincere thanks to all REFSQ participants who participated enthusiastically in the Alive Empirical Study, in the Online Questionnaires, and in the lively discussions at the Empirical Fair. This participation made the Empirical Track a real success.

Programm Committee

Sebastian Adam	Fraunhofer IESE, Germany
Ian Alexander	Scenario Plus, UK
Claudia P. Ayala	Technical University of Catalunya, Spain
Maya Daneva	University of Twente, The Netherlands
Deepak Dhungana	Siemens AG, Austria
Christof Ebert	Vector, Germany
Samuel Fricker	Blekinge Institute of Technology, Sweden
Thomas Gehrke	Siemens Rail Automation, Germany
Martin Herget	Siemens Corporate Technology, Germany
Andrea Herrmann	Infoman AG, Germany
Frank Houdek	Daimler AG, Germany
James Hulgan	Seilevel Inc, USA
Andreas Jedlitschka	Fraunhofer IESE, Germany
Eric Knauss	University of Victoria, Canada
Anne Koziolek	University of Zurich, Switzerland
Søren Lauesen	IT-University of Copenhagen, Denmark
Nazim H. Madhavji	University of Western Ontario, Canada
Luisa Mich	University of Trento, Italy
Gil Regev	EPFL and Itecor, Switzerland
Björn Regnell	Lund University, Sweden
Mehrdad Sabetzadeh	Simula Research Laboratory, Norway
Victoria Sakhnini	University of Waterloo, Canada
Camille Salinesi	Univ. Paris 1 – Panthéon Sorbonne, France
Erik Simmons	Intel, USA
Karen Smiley	ABB Corporation, USA
Roel Wieringa	University of Twente, Netherlands

6 Alive Empirical Study

Alive Empirical Study Programme

Proposal of a Quasi-Experiment for Studying the Effect of Experience on Elicitation Effectiveness 155
Alejandrina M. Aranda, Oscar Dieste and Natalia Juristo

Proposal of a Quasi-Experiment for Studying the Effect of Experience on Elicitation Effectiveness

Alejandrina M. Aranda, Oscar Dieste, Natalia Juristo

Universidad Politécnica de Madrid, Facultad de Informática, Campus de Montegancedo, 28660 Boadilla del Monte, Spain

am.aranda@alumnos.upm.es, {odieste, natalia}@fi.upm.es

Abstract. We plan to perform a quasi experiment to evaluate the effect of experience on requirements elicitation. Researchers will play the role of customers, whereas participants will perform the role of analysts. Analysts will hold a 60 minute interview and will then be given 25 minutes to write up a report of their findings. Participant effectiveness will be compared with available data series on the effectiveness of novice analysts that we have collected previously.

Keywords. Elicitation, analyst effectiveness, experience, quasi experiment

1 Introduction

One critical success factor in requirements engineering (RE) is having a good analyst [10]. The influence of different analyst characteristics on elicitation effectiveness has been researched empirically [4]. The most commonly examined aspect is experience [1], [7], [8], [9].

Results are controversial, as they tend to contradict RE folklore. Marakas and Elam [7] found that experienced analysts are only marginally better than novices. Pitts and Browne [9] report that analyst experience does not influence the quantity, breadth or depth of the requirements. Niknafs and Berry [8] conclude that experience has a negative influence, that is, experienced subjects are slightly less effective than inexperienced subjects. Finally, Agarwal and Tanniru [1] find that experienced subjects were slightly (but not significantly) better than inexperienced subjects.

We have run several studies as part of this research line, described in Section 2. However, the students that have participated as experimental subjects tend to be rather inexperienced. The aim of the study designed in this proposal, described from Section 3 onwards, is to gather data on subjects, who, like REFSQ'13 participants, are highly experienced. This study will benefit the RE community, as it will help to improve our understanding of the experience/effectiveness relationship. Additionally, we believe that the low effectiveness of expert analysts observed to date is due to factors other than experience, such as problem knowledge or *Einstellung* effects [2]. The proposed study will also examine such possible relationships.

2 Our Previous Studies

We have run three quasi-experiments (which we will call Q07, Q09 and Q11, according to the years in which they were run) with the aim of studying the influence of experience on elicitation process effectiveness. Quasi-experiments are conducted when subjects cannot be randomly assigned to an experimental condition, like subject experience, for example.

We have collected data from a total of 31 Universidad Politécnica de Madrid (UPM) software engineering postgraduate students. In all cases, students acted as analysts, gathering information about a fictional software system. Students used the open interview as an elicitation technique, as this is a straightforward technique for analysts, which is also widely used in practice [11]. Analyst effectiveness was calculated as the percentage of correctly identified and reported problem elements (e.g. concepts, requirements, etc.). Correctness is defined as the correspondence between the elicited problem elements and a gold standard established previously.

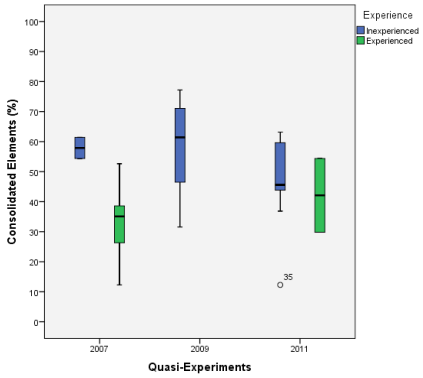

Fig. 1. Historical series of results about the experienced/effectiveness relationship

The student population participating in Q07 was composed of subjects with different levels of experience (from 0 to 6 years in requirements activities). Again the relationship between experience and effectiveness, shown in **Fig. 1**, is contrary to RE folklore and resembles Niknafs et al.'s findings [8].

The experimental population used in Q07 is too small to be able to state for sure that experience has a negative impact. In order to gather more data, we ran replications Q09 and Q11. Unfortunately, the 2007 UPM postgraduate program targeted professionals, whereas students were recruited mostly from graduate courses as of 2008. On this ground, Q09 and Q11 students had hardly any requirements experience, and, consequently, it was not generally possible to compare novices and experts directly. However, we were able to build a data series from Q09 and Q11, which we can supplement with data from other sources. The data series is also illustrated in **Fig. 1**. REFSQ'13 provides a unique opportunity for gathering the data that we require.

3 Description of the Proposed Study

In line with existing literature, the working hypothesis is that *there is no relationship between analyst experience and elicitation process effectiveness* (i.e. novice and expert analysts are equally effective). In order to test this hypothesis, we propose to run a quasi-experiment similar to Q07/Q09/Q11, whose results could be combined with the existing data.

The study is composed of three tasks, as shown in **Fig. 2**. A researcher will play the role of customer, whereas participants will perform the role of analysts. Analysts will study the same software system used in the previous quasi-experiments. The system domain will not be announced until the start of the elicitation session to stop analysts from doing any preparation that might affect their effectiveness.

The quasi-experiment will conclude with a 5-minute questionnaire. This questionnaire will contain questions about participant qualifications and knowledge in order to identify any variables potentially modifying analyst experience and effectiveness.

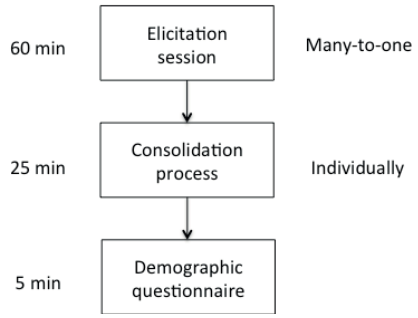

Fig. 2. Quasi experiment's activities for REFSQ'13

There are no restrictions regarding the participants' experience type or level. In order to escalate the quasi-experiment to the large number of attendees to REFSQ'13, the interviews will not be carried out individually (1 role-played customer and 1 analyst), but many-to-one (1 role-played customer and all analysts together). While this elicitation procedure is different than the typical 1:1 interview, it provides valuable data about the analyst's comprehension ability (as opposed to information extraction/capture), which is presumably one of the key factors affecting effectiveness. Furthermore, these data is directly comparable to the Q11 quasi-experiment, which applied a similar methodology.

No special equipment is required for running the quasi-experiment. Preliminary results will be available the day after the quasi-experiment. The data will be mapped out as a box-plot and added to the series illustrated in **Fig. 1**. No information by which specific participants can be identified will be published.

4 Threats to Validity

There are three main threats to the validity of the proposed study: (1) the customer participating in the interview sessions is fictional. Therefore, he will, to some (small or large) extent, be different from real customers; (2) the intended software system is not real; and, finally, (3) elicitation is conducted in a single session with a time limit.

The measures taken to mitigate these threats are: (1) we have carefully studied the target software system and played the role of customers in interview sessions in the context of laboratory experiments using the same system repeated times; (2) the system used during the experiment is based on an existing, real software system; and (3) most of the original system's complexity has been removed to make it easy to understand completely in a short (maximum 1-hour) period.

5 References

1. Agarwal, R., & Tanniru, M. R.: Knowledge Acquisition using Structured Interviewing: An Empirical Investigation. Journal of Management Information Systems, 7 (1990) 123 141
2. Anderson, J.: Cognitive psychology and its implications. 5th ed. Worth Publishers (1999)
3. Dieste, O., Griman, A., Juristo, N.: Developing Search Strategies for Detecting Relevant Experiments. Empirical Software Engineering, 14 (2009) 513 539
4. Dieste, O., & Juristo, N.: Systematic Review and Aggregation of Empirical Studies on Elicitation Techniques. IEEE Transactions on Software Engineering, (2011) 304
5. Juristo, N., & Moreno, A. M.: An Adaptation of Experimental Design to the Empirical Validation of Software Engineering Theories. 23nd Annual NASA Software Engineering Workshop (1998)
6. Juristo, N. M. A. M.: Basics of Software Engineering Experimentation. Kluwer Academic Publishers (2001)
7. Marakas, G. M., & Elam, J. J.: Semantic Structuring in Analyst and Representation of Facts in Requirements Analysis. Information Systems Research, 9 (1998) 37 63
8. Niknafs, A., & Berry, D. M.: The Impact of Domain Knowledge on the Effectiveness of Requirements Idea Generation during Requirements Elicitation. 20th IEEE International Requirements Engineering Conference (RE), (2012) 181 190
9. Pitts, M. G., & Browne, G. J.: Stopping Behavior of Systems Analysts during Information Requirements Elicitation. Journal of Management Information Systems, 21 (2004) 203 226
10. Schenk, K. D., Vitalari, N. P., Davis, K. S.: Differences between Novice and Expert Systems Analysts: What do we Know and what do we do? J. Manage. Inf. Syst., 15 (1998) 9 50
11. Zowghi, D., & Coulin, C.: Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In: Aurum, A. and Wohlin, C. (eds.), pp. 19 46. Springer Berlin Heidelberg (2005)

Submitters' experience: Natalia Juristo has been working in the field of experimentation since 1998 (e.g. [5]). She has written one of the reference books on experimentation in software engineering [6]. Oscar Dieste has been performing experiments in RE since 2007. They are also experienced in systematic reviewing and aggregation (e.g. [3]). Alejandrina Aranda is researching her PhD in RE at the Universidad Politécnica de Madrid.

7 Online Questionnaires

Online Questionnaires Programme

Using a Pattern Catalogue in Requirements Engineering Activities 161
Cristina Palomares, Xavier Franch and Carme Quer

A Survey on Lessons Learnt in Requirements Engineering 167
Ibtehal Noorwali and Nazim H. Madhavji

Do Mobile-App Users Care about Privacy? 171
Mariano Ceccato, Alessandro Marchetto, Anna Perini, Angelo Susi and Liria Veronesi

Prioritization of Security Requirements for Cloud Computing 177
Georg Herzwurm, Norman Pelzl, Benedikt Krams and Sixten Schockert

Online Questionnaire: Using a Pattern Catalogue in Requirements Engineering Activities

Cristina Palomares[1], Xavier Franch[1], Carme Quer[1]

[1] GESSI Research Group, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
{cpalomares, cquer, franch}@essi.upc.edu

Abstract. This paper presents a proposal of online questionnaire whose main goals are: to know current requirements engineering practices related with es sential aspects for the definition of requirement patterns and to analyze the cur rent and potential use of patterns in industry. The target audience of the ques tionnaire is any practitioner or academic with different levels of experience on requirement engineering.

Keywords: Requirement engineering practices, Requirement engineering prob lems, Requirement reuse; Requirement Patterns.

1 Introduction. Research Questions

Background. Requirements reuse has been proposed as a key asset for requirement engineers to efficiently elicit, validate and document software requirements and as a consequence, obtain software requirement specifications (SRS) of better quality through more effective engineering processes [1].

- *Context*. In the PABRE framework [2][3], the GESSI@UPC and SSI@TUDOR groups have adopted software requirement patterns (SRP) as approach to reuse [4]. PABRE includes a catalogue composed of 29 Non-Functional SRP (NF-SRP) and 37 [5] Non-Technical SRP (NT-SRP) [6]. The purpose of the catalogue is to suit the needs of the requirements elicitation and documentation processes carried out by TUDOR and associated IT consultants. Although the experience so far has been satisfactory overall, we have not had the chance yet to conduct any survey due to lack of population. We consider this type of research essential at the current moment of investigation.
- *Goal*. Using GQM, the goal of our online questionnaire can be defined as follows: to analyze (*purpose*) the benefits of the use of SRP (*issue*) in the requirements elicitation and documentation activities (*object*) from the perspective of require-ments engineering researchers and practitioners (*viewpoint*). The survey will not focus specifically in the PABRE catalogue, to make the proposal more appealing to the community and results more generalizable.

Research Questions. We operationalize the goal into 3 research questions (RQ):

- RQ1: What type of non-functional and non-technical requirements (NFR, NTR respectively) are more critical in a requirements reuse context? Rationale: we have observed that NFR and NTR appearing in SRS produced by TUDOR and associ-

ated IT consultants are recurrent. We expect this situation to occur in other contexts and, under this hypothesis, we want to know which types are critical.

- RQ2: Is requirements reuse a usual practice in current RE processes? *Rationale*: we need to know how fundamental the problem of reuse is for the respondent, and which reuse strategies are currently in place. This knowledge will help to contextualize the rest of information gathered in the interview.
- RQ3: Could the existence of a catalogue of SRP help improving the effectiveness of requirement elicitation and documentation? Rationale: using the catalogue, requirements are usually not built from scratch; instead, the catalogue yields to a process that guides the engineer by giving recommendations suggesting information, etc. Likewise, a high-quality catalogue is expected to include SRP templates that have been designed by using a uniform style, a glossary, and determining properties like dependencies upon other SRP. We expect these characteristics to have a positive effect on the elicitation processes conducted and the SRS produced.

2 Questionnaire Design

In general, the questionnaire will offer multi-choice questions, using a 5-value Likert scale when qualitative answers are required, and will be composed of six parts (see Fig. 1). The first three parts will be used to know the context and experience of the respondents, and general aspects about requirements engineering practices and problems they came across. The other three parts are respectively related to the three research questions.

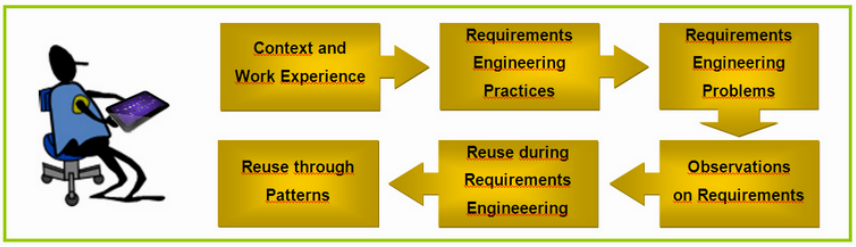

Fig. 1. Survey Flow

Context and Work Experience. The first one will include the usual questions about the context and experience of the respondent to filter and group participants during analysis. The questions to the participants will change depending on where his/her experience as requirements engineer comes from. Specifically, we will distinguish among (see Fig.2): Industry or academy with a significant experience in industry projects; Academy with some knowledge of industry practices; Academy without any exposure to industry.

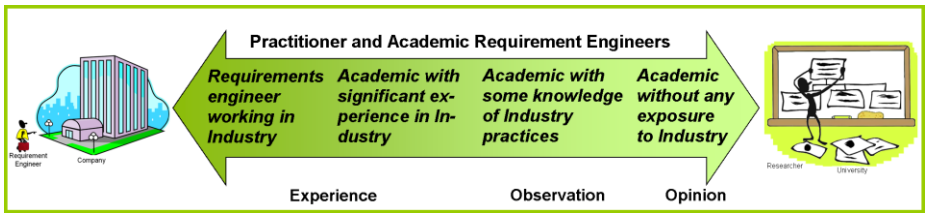

Fig. 2. Target Audience

Requirements Engineering Practices. The second part will characterise the general aspects and requirement engineering practices of projects in which the respondents have been involved in (including language used to express requirements).

Requirements Engineering Problems. The third part will try to state the commonality of different problems during elicitation and specification processes of projects where the respondents have been involved (contradictory needs, ambiguity, lack of traceability,…).

Observations on Requirements. The questions in this part will be the base for answering RQ1.

- How recurrent are NFR&NTR in the projects conducted by respondents?
- For a given classification schema based on the ISO 25010 quality model with some NTR categories added from our own catalogues, we will ask how often do they appear in their projects. The respondent will be allowed to add additional categories if s/he considers there is any missing.
- According to the classification schema, which are the 3 types of NFR&NTR that are more challenging to deal with? Which ones are the most prioritary?

Reuse during Requirements Engineering. For answering RQ2, we will ask a few general questions:

- How recurrent are requirements from one project to other?
- Is requirement reuse currently seen as a challenge in RE processes?
- How is requirements reuse implemented in the respondents' projects?

Reuse through Patterns. In case of RQ3 we will ask about the benefits that requirements reuse through SRP could bring to companies and possible barriers to adoption. Some questions will be:

- Which of the problems observed in elicitation and specification could be ameliorated by the existence of an SRP catalogue?
- What would be necessary in order to provide such a benefit (e.g., tool support, training, well-defined method)?
- What would be the most important barriers to the successful adoption of such approach?

Before finishing the questionnaire, the respondents will be able to state any clarification or comment about his/her answers or the questionnaire, and also to ask the feedback about the questionnaire results analysis being sent to them.

3 Venues of Publicity and Intended Procedure

Availability. The online questionnaire will be available as a LimeSurvey. It will be installed in the GESSI@UPC server and accessible through the PABRE website [4]. It will be open for REFSQ'13, and remain accessible a few weeks after the conference to get late responses and allow some ripple effect from REFSQ'13 attendees.

Resources. Assuming there will be WiFi connection available, no extra resources are required. It would be nice to have some workstation in the coffee-break area available for attendees answering the different questionnaires that may be offered.

Publicity. Before the conference, in addition to REFSQ'13 publicity, we plan to make publicity using our local networks and selected (to avoid spam) distribution lists like re-online. We will intensively use our own resources (GESSI twitter, @gessi upc; GESSI LinkedIn group; GESSI and PABRE webpages). We will provide the organization with some publicity pack, both digital and on print (e.g., a bookmark). We will prepare a poster to announce the questionnaire at the coffee-break space if the organization allows so.

Data analysis. We will apply descriptive statistics and content analysis (for questions with results in free text). Other techniques we plan to apply are: correlation analysis; and cluster analysis. Analysis will be done by the full set of authors.

Results. We commit to offer an overview of the results during the conference at the slot requested by organizers. A detailed analysis will be published as technical report in GESSI@UPC and announced to the community using the same channels as for publicity. Complete results will be sent to the participants.

4 Participation Subjects and Benefits

Subjects. The questionnaire will be open to all participants in REFSQ'13, both researchers and practitioners. We assume that all of them have good knowledge about requirements elicitation and documentation. There is no maximum number of participants. For the minimum, we think that 40 participants in the conference (that may lead to over 30 valid results) would be admissible, since we plan to distribute the questionnaire in the community after REFSQ'13 to get more responses. It is important to know that the best paper award in REFSQ'12 went to a research paper based on observations made from a questionnaire with 39 responses, which points out that our target can be considered appropriate in the context.

Benefits. As far as we know, this will be the first survey centered in requirements reuse conducted in the RE community. For the REFSQ'13 community, a benefit is that the partial results will be drafted and presented at the end of the conference, and a research report will be published with a detailed analysis once the study finishes. Individual participants may benefit as part of this community. Also, we will license the detailed statistics of the individual questions to who may be interested under a Creative Commons Licence. Also, we will offer them the possibility of duplicating the LimeSurvey space for participants that want to replicate the study themselves.

5 Threats to validity and mitigation actions

We focus on the risk we consider more likely to occur.

Varied profile of attendees. REFSQ attendees may be practitioners or researchers. Having two profiles so radically different is a risk in the sense that questions may be inappropriate for one type. To mitigate this risk, we will refine the questions outline in Section 2 differently when it is required. As an opportunity, we may consider a fourth RQ meaning: are expectations that researchers have on SRP fulfilled in reality? Going further, the nature of companies, research centres and projects may be very different and jeopardise generalization of results. At the extreme, quantitative analysis may become difficult and may require accompanying qualitative reasoning.

Willingness to participate. REFSQ'13 attendees have usually lot of networking and activities to do and a threat is that they do not find the moment to fill the questionnaire. Mitigation actions are: networking ourselves; bringing hardcopies to allow attendees to answer outside the REFSQ'13 premises; asking REFSQ'13 organization for support to publicity (e.g., whiteboards to post questionnaire announcements).

Questionnaire design. As any online questionnaire, participants could misunderstand some of the questions or values suggested as answers, miss values as answers in multiple-answer questions or find the questionnaire too heavy to be answered. We will mitigate these threats by: accompanying the questions with a glossary of terms accessible through hyperlinks; adding whenever necessary text fields for clarification or adding missing values; exhaustively pilot the questionnaire to ensure it does not take more than 30 minutes so that respondents do not give up; order an English revision by a native speaker.

External validity. Since participants in our questionnaire are not selected randomly it is not possible to warrant that its results correspond with good confidence to the results that would be obtained conducting the same survey to the whole RE population. Mitigating this risk (inherent to any survey ran during REFSQ) needs some time in order to replicate the survey in other contexts and then check if it happens.

6 Past Empirical Studies Performed

The GESSI@UPC's paper authors have well-proven abilities in conducting empirical research. Results have been published in different venues, including REFSQ. The topics and the research instruments are manifold, as can be seen in Table 1.

Table 1. Empirical research in which GESSI@UPC's authors have been involved.

Topic	Instrument	Venue	Collaboration with
Information quality in OTS selection	Questionnaire	ICCBSS'08	
Challenges in OSS industry	Interviews	OSS'09	NTNU
OSS selection practices	Interviews	OSS'11, JSS	NTNU
Requirements in OSS selection	Interviews	REFSQ'12	NTNU, U. Lund
OSS integration and communities	Interviews	OSS'12	NTNU, U. Lund, U. København
Perception on NFR types' importance	Questionnaire	REFSQ'10	
How architects deal with NFRs	Interviews	RE'12	INRIA

Acknowledgements. This work has been partially supported by the Spanish project TIN 2010-19130-C02-01

References

1. Lam, W., McDermid, J.A., Vickers, A.J.: Ten steps towards systematic requirements reuse. Requirements Engineering J.. 6,15, 6 10 (1997).
2. Franch, X., Palomares, C., Quer, C., Renault, S., De Lazzer, F.: A Metamodel for Software Requirement Patterns. In: Int. Work. Conf. on Requirements Engineering: Foundation for Software Quality (REFSQ), pp 85 90. Springer (2010).
3. Renault, S., Mendez, O., Franch, X., Quer, C.: A Pattern based Method for building Re quirements Documents in Call for tender Processes. In: Int, J. of Computer Science & Ap plications. 6, 5, 175 202 (2009).
4. PABRE website, http://www.upc.edu/gessi/PABRE/Patterns.html
5. Franch, X., Quer, C., Renault, S., Guerlain, C., Palomares, C.: Constructing and Using Software Requirements Patterns. In: Managing Requirements Knowledge, Maalej, W., Thu rimella, A.K. (Eds.). Springer (2013).
6. Palomares, C., Quer, C., Franch, X., Guerlain, C.; Renault, S.: A catalogue of non technical Requirement Patterns. In: Sec. Int. Work. on Requirements Patterns (RePa), pp 6. IEEE Press (2012).

A Survey on Lessons Learnt in Requirements Engineering: A Proposal for the Empirical Track at REFSQ'13

Ibtehal Noorwali and Nazim H. Madhavji

Department of Computer Science, University of Western Ontario, London, Canada
{inoorwal@uwo.ca, madhavji@gmail.com }

1 Study Type

The proposed study is an online survey.

2 Survey Goals

- To elicit lessons learnt [1] in RE from significant software intensive projects (not classroom projects) from participants attending the REFSQ13 conference.
- To synthesize a readily accessible body of knowledge of lessons learnt in RE elicited from the REFSQ13 community.

3 Background Context

- From an industry survey [7] we conducted on the use of RE lessons in software projects, over 70% of the respondents stated that they seldom use RE lessons in the RE process, though 85% of these would use such lessons if readily available.
- Our observation, however, is that, RE lessons are scattered, mainly implicitly, in the literature and practice, which, obviously, does not help the described situation.
- Approximately 90% of the survey participants stated that not utilising RE lessons has significant negative impact on product quality, productivity, project delays and cost overruns.
- The non-RE literature on lessons learnt can be roughly categorized into (i) discovering and sharing lessons learnt (e.g., [2, 3]) and (ii) technologies to support lessons learnt (e.g., [1, 8, 9]).

[1] Lesson Learnt: A successful experience that encourages the same/similar behaviour in a similar situation in the future to achieve the same/similar observed results; or, an unpleasant experience that requires different behaviour in a similar situation in the future to avoid the observed results.

- In RE, little attention has been paid to lessons learnt; while some literature describes lessons learnt explicitly [4, 5], much of it describes lessons implicitly in the textual description [6]. This makes it difficult to utilise lessons in RE practice.
- In order to ameliorate this situation, we propose to conduct a survey for the purpose of eliciting lessons learnt in RE.

4 Description of Procedure

- The survey consists of four demographic questions and a template that consists of text areas for the information about the lesson learnt and an area for the lesson text.
- This is an anonymous survey.
- The web link to the online survey will be provided to the participants.
- The online survey software surveygizmo.com (widely-used) will be used to conduct the survey and gather results.
- Participants can take the survey at any time they find convenient.
- The survey is not anticipated to exceed 30 minutes. To fill one template (one lesson) is anticipated to take only a few minutes (say, approximately 5 minutes). Participants can fill more than one template.
- After the survey is closed, the submitted lessons will be analysed for completeness and relevance.
- The synthesized set of lessons gathered will be publicized according to the REFSQ call for proposal requirements.
- A link to the current version of the survey is:
  http://edu.surveygizmo.com/s3/1131891/A-Survey-of-Lessons- Learnt-in-Requirements-Engineering

5 Benefits of the Results to the RE Community

- Eliciting lessons learnt from REFSQ attendees is anticipated to create a body of lessons, which can be readily available to the RE community.
- Each lesson has a set of attributes (e.g., application domain, context of the lesson, related RE phase, source, project size, etc.). Thus, this body of lessons learnt can be analysed from specific viewpoints pertaining to the desired attributes.
- The viewpoint-based analysis of the body of lessons would highlight the relatively scarcely and densely populated RE areas. This analysis is anticipated to promulgate research in the scarce areas and improve practices in the denser areas of RE.
- A tool can then be built to support and operationalise viewpoint-based analyses. Example operations include: add, edit, delete, and search for lessons; clustering a set of related lessons for solving specific problems (e.g., hazard analysis of train brakes); sharing lessons among desired individuals; finding problem-specific lessons; etc. This tool is anticipated to facilitate the use of

lessons in projects, which in turn, would likely have a positive impact on cost, quality, error avoidance, and time.

6 Benefits of the Participants

- Upon request, the participants will receive the results of the survey free of charge.
- Participants will have a chance to (re)think about the RE lessons that they may have learnt in the past, making it explicit for them.
- Participants will know that there are others in the RE community who are concerned about RE lessons; that they are not alone. This could motivate them to form and join special interest groups, organize workshops, etc., which in turn, has RE community benefits.

7 Minimum and Maximum Number of Subjects

- There is no minimum and maximum number of subjects in this survey.

8 Profile of the Intended Subjects

- Anyone with practical RE experience in real-world (non-classroom) projects. All areas of RE are acceptable.

9 Threats to Validity

- Construct validity: because this is an online survey, and not an interview, a participant could interpret our definition differently from what we had meant. Hence, the data gathered could have noise in it and could cause analysis problems. This threat exists because we guarantee participant anonymity, so we cannot get back to them for clarification.
- Conclusion validity: due to the threat to construct validity, there could be a threat to conclusion validity.

10 Record of Past Empirical Studies

- Noorwali, I., Madhavji, N.H.: A Survey of Lessons Learnt in Requirements Engineering. Technical Report No. 750, Dept. of Computer Science, University of Western Ontario (2012)
  http://edu.surveygizmo.com/s3/961128/A-Survey-of-Lessons-Learned-in-Requirements-Engineering
- Numerous other empirical studies performed by the second author and his team.

11 Required Venues

– The survey will be publicized and distributed via emails, REFSQ13 conference website (if permitted), and hardcopies of the announcement.
– Participants need to have access to a smart phone, tablet, or computer, and the Internet in order to complete and submit the survey.

References

1. Abdel-Hamid, T.K., Madnickn, S.E.: The Elusive Silver Lining: How we Fail to Learn from Software Development Failures. J. MIT Sloan Management Review. 32, 39–48 (1990)
2. Basili, V.R., McGarry, F.E., Pajerski, R., Zelkowitz, M.V.: Lessons Learned from 25 Years of Process Improvement: The Rise and Fall of the NASA Software Engineering Laboratory. In: International Conference on Software Engineering, pp. 69–79. ACM (2002)
3. Boehm, B.: A View of 20th and 21st Century Software Engineering. In: In: International Conference on Software Engineering, pp. 12-29. ACM, Shanghai (2006)
4. Damian, D.: Stakeholders in Global Requirements Engineering: Lessons Learned from Practice. IEEE Software Journal. 24(2), 2127 (2007)
5. Daneva, M.: ERP Requirements Engineering Practice: Lessons Learned. IEEE Software Journal. 21(2), 2633 (2004)
6. Ebert, C.: Understanding the Product Life Cycle: Four Key Requirements Engineering Techniques. IEEE Software Journal. 23(3),1925 (2006)
7. Noorwali, I., Madhavji, N.H.: A Survey of Lessons Learnt in Requirements Engineering. Technical Report No. 750, Dept. of Computer Science, University of Western Ontario (2012)
8. Sary, C. and Mackey, W.: A Case-Based Reasoning Approach for the Access and Reuse of Lessons Learned. In: Fifth Annual Symposium of the National Council on Systems Engineering, pp. 249256. St.Louis (1995)
9. Vandeville, J.V., Shaikh, M.A.: A Structured Approximate Reasoning-Based Approach for GatheringLessons Learned Information from System Development Projects. J. Systems Engineering. 2(4), 242247 (1999)

Do Mobile-App Users Care about Privacy?

Mariano Ceccato, Alessandro Marchetto, Anna Perini, Angelo Susi, and Liria Veronesi

Fondazione Bruno Kessler, Trento, Italy.
{ceccato,marchetto,perini,susi,veronesi}@fbk.eu
Tel: +39.0461.314.577 Tel: +39.0461.314.330 Tel: +39.0461.314.344

Abstract. Smartphone users can easily install mobile apps on their mobile devices with just a single click on the "accept" button of the privacy conditions review page. Indeed, privacy is a key requirement of mobile apps. This motivates us to propose an empirical study aiming at investigating how users perceive and evaluate privacy in this context.

In this proposal for Empirical Studies at REFSQ (ESR), we intend use an on-line questionnaire to study how mobile-app's users perceive privacy issues.

1 Problem definition

Nowadays mobile-apps can be easily installed on mobile devices (smartphones and tablets) by their end-users upon a simple click on the "accept" button, which comes at the end of a 1-screen description of main functionalities of the app, including requests of access to device' components or user's data. Possible privacy risks are sometimes made clearly explicit — e.g. "Malicious apps may use this to erase or modify your contact data" is used to describe the risk associated to enabling an app to access the user contact data —, sometime they are left implicit, — e.g. "An app with this permission, when a call is active, can determine the phone number, and serial number of the caller's phone" that brings as consequence that phone state and caller identity can be detected—. That is, there can be strong implications between app's features and user privacy. The current solution adopted by app providers seems to let the end-user decide whether to accept risks and potential privacy violations. However, it is questionable if users really understand (or even care about) privacy implications when installing new apps. Nonetheless the variety of apps is growing very rapidly (nowadays you can find an app for almost everything), as well as their diffusion, resembling virus spread out phenomenon.

This motivates us to investigate on what is the actual process that users adopt when deciding to install new apps, in-spite of, or according to, their attitude to privacy. One step in our research will consist in investigating if end-users care about privacy issues when selecting and installing apps on their mobile device, and how do they decide whether to install them.

2 The On-Line Questionnaire

2.1 Objective

Our study is driven by the two following main research questions:

RQ1 Do end-users *care about privacy* issues when selecting and installing apps on their mobile devices?

RQ2 *How* do end-users *evaluate* app features implications on their privacy, when selecting and installing apps on their mobile devices?

The questionnaire has an *exploratory* purpose [1].
Proper hypotheses and observations will be formulated according to the collected data, and support further investigation.

2.2 Procedure

The on-line questionnaire will recall concrete app-installation scenarios, with reference to popular app categories (e.g., car navigators or restaurant advisers) with the aim to motivate a subject to revisit her/his own experience as app-user. Then, open and closed questions will be formulated to ask participants to describe their concrete experiences when selecting and installing these apps, for instance if/why apps have been installed and if there were any potential generic *drawback*.

After collecting quantitative data with closed questions, free feedback will be solicited to collect additional qualitative data. Closed questions with reference to $RQ1$ will check, for example, weather users paid attention to privacy implications which could have emerged during the installation of an app.

With reference to $RQ2$, closed questions will concern how users considered possible requests of access to device' components or user's data, when installing an app. Quantitative feedback will be collected using ordinal scales.

The last part of the questionnaire will be devoted to profile the participant from the point of view of her/his skill, expertise, experience in ICT and apps, her/his attitude to privacy, e.g. her/his exposure to social networks.

Open answers will be subject to grounded theory [2], while quantitative data collected with closed questions will be analyzed using descriptive statistics [3].

2.3 Subjects

We expect at least 15 participants to fill the questionnaire. We do not set any maximum number. The participants should be apps users, who select, download and use apps on their mobile device for their personal or daily life activities.

REFSQ participants can be considered as representative of expert in ICT, and particularly in requirements engineering for ICT. To get a more complete view of the phenomenon, the same questionnaire will be proposed also to different categories of users, e.g. Humanities research scholars.

2.4 Benefits of the Study

The main benefit for the *Requirements Engineering community* will be to get insights into app-users' awareness and concern about privacy issues. In particular, considering different types of users and different kind of functionalities, the aim is that of identifying if there exists any relations between attitude to privacy and user characteristics. The scientific community seems to lack this type of studies with respect to mobile apps.

As part of the requirement engineering community, *participant to the questi onnaire* will be able to access to the results of our study and possibly reuse them in their research or industrial initiatives.

2.5 Threats to Validity

Performing the questionnaire at REFSQ might originate a threat to the *external validity* of the study, limiting the generalization of our findings. To mitigate this threat, we plan to replicate the study with different categories of subjects, e.g. experts in software system security; and non expert in ICT engineering.

Moreover, a threat to *construct validity* can originate if subjects will try to guess our study hypotheses. In fact, participants might be biased by their professional interests and their answers may come from their knowledge in the field of apps engineering, rather than from their actual experience as apps users. In order to mitigate this threat, we carefully designed the questionnaire by recalling concrete usage experiences, avoiding to use words like "privacy" (so the title of our survey will be different w.r.t. the actual one used in this proposal).

We will mitigate the threat to the *conclusion validity* of our findings by using objective statistical tests.

3 Previous Studies

The submitters of this proposal have a long experience in empirical studies on many aspects of software engineering and social science.

Software Engineering. Among the most relevant experiments in this area are: empirical evaluation of requirements modelling with the Tropos4AS framework [4]; controlled experiments to evaluate requirements model understanding, comparing *Use Case* vs. *Tropos* models [5]; evaluating tool-supported requirements prioritization: a controlled experiment comparing AHP and CBRank [6]; evaluating the effectiveness of using acceptance test to clarify change requirements [7] [8] [9] evaluating the effectiveness of stereotypes to support the comprehension of UML diagrams [10] [11] [12] [13]; evaluating the improved understandability of new programming languages with respect to software maintenance [14]; evaluating how the different types of test cases affect debugging [15]; evaluating the level of protection offered by source code obfuscation against malicious code tampering [16].

Social Science. Quantitative empirical study techniques have been exploited to

investigate the relationship between Humanistic and Scientific research domains, as perceived by researchers in the two domains [17]. Social network analysis techniques and quantitative empirical study approaches have been adopted to investigate about social cohesion [18], and about the role of gender in the context of Humanistic and Scientific research [19], respectively.

4 Publicizing the Study

In the study, quality of answers and quantity of participants are two important factors. In the case of the experiment at REFSQ we will exploit the mailing lists of the authors, and conference attendants, to stimulate participation via appropriate mail messages. Other groups of participants will be involved exploiting ongoing national and international projects as well as other ongoing initiatives involving our research center and other local organizations (e.g. secondary schools, university campus).

5 Equipment

An Internet connection is required to fill the on-line questionnaire. No other special equipment is required.

References

1. Babbie, E.R.: The practice of social research. Wadsworth Publishing Company (2012)
2. Oppenheim, A.N.: Questionnaire Design, Interviewing and Attitude Measurement. Pinter, London (1992)
3. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A.: Experimentation in Software Engineering - An Introduction. Kluwer Academic Publishers (2000)
4. Morandini, M., Perini, A., Marchetto, A.: Empirical evaluation of tropos4as modelling. In de Castro, J.B., Franch, X., Mylopoulos, J., Yu, E.S.K., eds.: iStar. Volume 766 of CEUR Workshop Proceedings., CEUR-WS.org (2011) 14–19
5. Hadar, I., Kuflik, T., Perini, A., Reinhartz-Berger, I., Ricca, F., Susi, A.: An empirical study of requirements model understanding: *Use Case* vs. *tropos* models. In: SAC. (2010) 2324–2329
6. Perini, A., Ricca, F., Susi, A.: Tool-supported requirements prioritization: Comparing the ahp and cbrank methods. Information & Software Technology **51**(6) (2009) 1021–1032
7. Ricca, F., Torchiano, M., Di Penta, M., Ceccato, M., Tonella, P.: Using acceptance tests as a support for clarifying requirements: A series of experiments. Information and Software Technology **51**(2) (2009) 270 – 283
8. Ricca, F., Di Penta, M., Torchiano, M., Tonella, P., Ceccato, M., Visaggio, C.A.: Are fit tables really talking? a series of experiments to understand whether fit tables are useful during evolution tasks. In: Proceedings of the 30th International Conference on Software Engineering (ICSE 2008), IEEE Computer Society (10-18 May 2008) 361–370

9. Ricca, F., Torchiano, M., Di Penta, M., Ceccato, M., Tonella, P.: The use of executable fit tables to support maintenance and evolution tasks. In: Proceedings of the Third International ERCIM Symposium on Software Evolution (Evol 2007). (October 2007) 83–92
10. Ricca, F., Di Penta, M., Torchiano, M., Tonella, P., Ceccato, M.: How developers' experience and ability influence web application comprehension tasks supported by uml stereotypes: A series of four experiments. Software Engineering, IEEE Transactions on **36**(1) (Jan.-Feb. 2010) 96–118
11. Ricca, F., Di Penta, M., Torchiano, M., Tonella, P., Ceccato, M.: How design notations affect the comprehension of web applications. Journal of Software Maintenance and Evolution: Research and Practice **19**(5) (2007) 339–359
12. Ricca, F., Di Penta, M., Torchiano, M., Tonella, P., Ceccato, M.: The role of experience and ability in comprehension tasks supported by uml stereotypes. In: Proceedings of the 29th International Conference on Software Engineering (ICSE 2007), IEEE Computer Society (20-26 May 2007) 375–384
13. Ricca, F., Di Penta, M., Torchiano, M., Tonella, P., Ceccato, M.: An empirical study on the usefulness of conallen's stereotypes in web application comprehension. In: Proc. of 8th IEEE International Symposium on Web Site Evolution (WSE 2006), IEEE Computer Society (September 2006) 58–68
14. Tonella, P., Ceccato, M.: Refactoring the aspectizable interfaces: an empirical assessment. IEEE Transactions on Software Engineering **31**(10) (October 2005) 819–832
15. Ceccato, M., Marchetto, A., Mariani, L., Nguyen, C.D., Tonella, P.: An empirical study about the effectiveness of debugging when random test cases are used. In: 2012 34th International Conference on Software Engineering (ICSE), IEEE Computer Society (2012) 452–462
16. Ceccato, M., Di Penta, M., Nagra, J., Falcarin, P., Ricca, F., Torchiano, M., Tonella, P.: The effectiveness of source code obfuscation: an experimental assessment. In: 17th IEEE International Conference on Program Comprehension (ICPC 2009), IEEE (May 2009) 178–187
17. Costa, P., Veronesi, L.: Le due culture in FBK: unindagine empirica. In: Annali di studi religiosi. EDB, Bologna (2010)
18. Veronesi, L.: Lo studio della coesione sociale attraverso un approccio di rete. In Di Nicola, S., Di Nicola, P., Stanzani, S., Tronca, L., et al., eds.: Forme e contenuti delle reti di sostegno. Il capitale sociale a Verona. FrancoAngeli (2010)
19. Chizzola, V., Veronesi, L.: Interdisciplinarity as a way to break the glass ceiling. the role of empathy. In: Proc. International Conference of Education, Research and Innovation, ICERI. (2011) 5856–5962

Prioritization of Security Requirements for Cloud Computing – Proposal for the Execution of an Online Questionnaire

Georg Herzwurm, Norman Pelzl, Benedikt Krams, Sixten Schockert

Department for Business Administration and Information Systems II, esp. Business Software, University of Stuttgart, Keplerstr. 17, Stuttgart, Germany,
{herzwurm|pelzl|krams|schockert}@wius.bwi.uni-stuttgart.de

1 Goals of the study and hypotheses

Cloud Computing (CC) – as a paradigm shift where e.g. customers need no longer invest in hardware, software or services – is often mentioned as a way to reduce operational and maintenance costs [6]. Trade shows like CeBIT 2012, which had CC as the keynote trend, still suggest an increasing importance of this topic in practice [1,2], although security issues seem to be the biggest obstacle to the adoption of CC [9]. However, proponents of CC consider security issues to be a myth [8].

The online questionnaire asks respondents to prioritize a proposed set of security requirements for Cloud Computing (CC) in order to obtain information about the relative importance of those requirements. To find out if an absolute increase in the importance of CC security requirements has occurred, the absolute importance of those security requirements with respect to other requirements (functional and other non-functional requirements) of CC will be examined too. From this, implications for Requirements Engineering (RE) methodology adoption for CC application development can be derived: are security requirements for CC as important as they are claimed to be, and if so, is there a necessity for the adoption of RE methods?

The *hypotheses* therefore are: (1) *Pairwise comparison of a dedicated set of security requirements leads to a prioritized set of security requirements and allows focusing on dedicated security requirements and proposed solutions within a CC project.* (2) *Absolute weighting of the increase of importance of a dedicated set of security requirements with respect to other functional and non-functional requirements does not demonstrate an increase in the importance of security requirements for CC.*

2 Detailed description of the intended procedure

According to the "CIA triad", confidentiality, integrity and availability are significant challenges for cloud computing [1]. But other security requirements, such as access control or attack harm detection, also exist [4]. A literature review of CC security

surveys led to the inclusion of the following security requirements for CC for evaluation in the questionnaire.

- Confidentiality
- Availability
- Control
- Benefit [1,3,4,6,9,10,12]
- Integrity
- Authentication
- Audit

Prioritization of the proposed set of security requirements will be done by pairwise comparison because of this method's applicability to measuring subjective criteria, as is done in the Analytical Hierarchy Process (AHP) [11]. Through the pairwise comparison, security requirements will be identified, allowing us to draw a conclusion about which security requirements for CC should be focused on. The Information Systems literature only provides some weak hints about which security requirements for CC should be implemented first; it is usually stated that all security requirements for CC are (equally) important [9]. This proposal does not take more requirements into consideration because of the curve progression which would lead to a huge increase in pairwise comparisons with each additional security requirement to evaluate:

$$N_n = N_{n-1} + (n-1)$$

In our proposed questionnaire the overall number of pairwise comparisons for $n = 7$ security requirements would be $N_7 = 21$.

We will use the "academic online research network" unipark software from Globalpark AG, which allows us to conduct several analyses of polar questions automatically as well as to analyze open-ended questions with comparatively little effort. The pairwise comparison will be polled within the tool with sliders like the ones illustrated below.

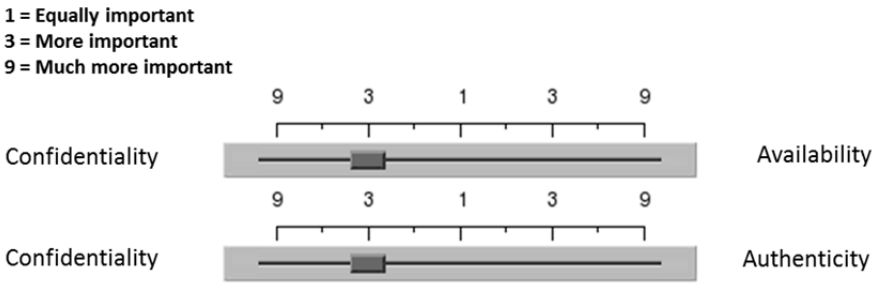

Fig. 1. Pairwise comparison with sliders in unipark

Using sliders and a 1, 3, 9 scale makes the pairwise comparison more comfortable for the participants. For example, using a matrix confronting all security requirements line by line and column by column would make the task more confusing, which might lead to some people aborting the questionnaire. Additionally, the software allows adding explanations for each security requirement if the curser is moved over a requirement ("mouse-over functionality").

To gain information about the absolute increase in the importance of CC security requirements, a five-point Likert scale will be used for weighting (1–much less important; 2–less important; 3–equally important; 4–more important; 5–much more important). The precise question would be: "Please conduct the weighting for the following CC security requirements by using the given scale". To examine the potential consequences for RE research, in the sense of necessary adoptions of RE methodology in CC application development, an open-ended question will also be given.

3 Benefits of the study

Besides the important benefit to academia of the insights this study would generate, we believe that industry might also profit by getting a clearer picture of which CC security requirements to focus on in the development of Cloud Computing applications. A prioritization of security requirements for CC might help to find better solutions to overcome the security issues mentioned above.

4 Profile of intended participants

The participants will be drawn from academia and industry. A prerequisite for participation is that they should be knowledgeable about Cloud Computing as well as have practical experience in using one or more cloud computing service models (SaaS, PaaS and IaaS) and/or deployment models (public, private, hybrid and community cloud). As the participants will be REFSQ attendees, we assume that a high rate of participants will use CC for business purposes.

5 Discussion of the threats to validity

By using a structured online questionnaire, pairwise comparison and a Likert scale, we aim at achieving reliable results. Nonetheless, the questionnaire will not be answered with a common understanding: each candidate will have a different background and individual level of knowledge. Furthermore, the level of expertise of those answering the open-ended question concerning future RE research will lead to subjective answers which can only serve to poll estimations and expectations.

Additional threats to validity might emerge if a critical mass of valid questionnaires is not reached. We will tackle this problem by keeping the questionnaire available online and continuing to promote it after the conference. This way we will reach more potential responders and obtain a higher number of valid answers. A draft analysis will be presented at the conference.

6 Record of empirical studies performed by the submitters

- Tauterat, T., Mautsch, L.O., Herzwurm, G.: Strategic Success Factors in Customization of Business Software. Software Business: Third International Conference, ICSOB 2012, Cambridge, MA, USA, June 2012, Proceedings (LNBIP 114), pp. 267 272, Springer (2012)
- Helferich, A., Herzwurm, G.: Softwaretechnische Ansätze für die Entwicklung flexibler Anwendungssysteme Ergebnisse einer explorativen Studie. In: Bichler, M. et.al. (eds.): Multikonferenz Wirtschaftsinformatik 2008, pp. 1741 1752 (2008)
- Herzwurm, G., Reiß, S., Schockert, S.: The support of Quality Function Development by the customer orientated evaluation of software tools. In: QFD Institute (eds.): Transactions from the Fifteenth Symposium on QFD and the Ninth International Symposium on QFD '03, Orlando, Florida, USA, pp. 139 158 (2003)
- Herzwurm, G. Schockert, S., Mellis, W., Ahlemeier, G.: Success Factors of QFD Projects. In: Ross L. Chapman, Robert Hunt (eds.): Proceedings of the World Innovation and Strategy Conference in Sydney, Australia, pp. 27 41 (1998)

7 Venues for publicizing the study and required infrastructure

As the authors of this paper will organize the fourth RePriCo workshop at REFSQ2013, this will naturally be the first venue for publicizing the questionnaire, as some attendees might only visit workshops but not the conference itself. We further encourage REFSQ2013 chairs to continue to promote the empirical track during the conference as they did in previous years. Flyers might help to remind some attendees about the online questionnaire, so a little space will be required for these throughout the conference. A poster will be made to promote the survey, and will require a partition wall.

The questionnaire will also be promoted outside REFSQ through the mailing lists of the authors. We also plan to use the mailing list of the special interest group Requirements Engineering within the German Informatics Society (Gesellschaft für Informatik e. V. – GI).

8 References

1. CeBIT: Keynote trends of CeBIT 2012. http://www.cebit.de/en/about the trade show/facts figures/about cebit 2013/end of show report cebit 2012/keynote trends of cebit 2012
2. BITKOM: Cloud Computing World at CeBit 2012. http://www.bitkom service.de/Events/cloud.aspx
3. Firesmith, D.: Specifying reusable security requirements, Journal of Object Technology, Vol. 3, No. 1, pp. 61 75 (2004)
4. Iankoulova, I., Daneva, M.: Cloud computing security requirements: A systematic review. In: Proceedings of Sixth International Conference on Research Challenges in Information Science, IEEE, pp. 1 7 (2012)
5. Joshi, A.B., Vijayan, B.S., Joshi, K.: Securing Cloud computing Environment against DDoS Attacks, IEEE, pp. 1 5 (2011)
6. Kilari, N., Sridaran, R.: A Survey on Security Threats for Cloud Computing. International Journal of Engineering Research & Technology, Vol 1, Issue 7 (2012)
7. Krutz, R.L., Vines, R.D.: Cloud security. A comprehensive guide to secure cloud computing. Indianapolis, Ind: Wiley Pub. (2010)

8. LeClaire, J.: Five IT Myths To Be Debunked in 2012. http://www.cio today.com/story. xhtml?story title Five IT Myths To Be Debunked in 2013&story id 0320013QJD4W
9. Popovic, K., Hocenski, Z.: Cloud computing security issues and challenges. Proceedings of the 33rd International Convention. pp. 344 349 (2010)
10. PricewaterhouseCoopers AG (ed.): Cloud Computing. Navigation in der Wolke (2011)
11. Saaty, T.L.: Decision making with the analytic hierarchy process. International Journal Services Sciences, Vol. 1, No. 1, pp.83 98 (2008)
12. Zhou, M., Zhang, R., Xie, W., Qian, W., Zhou, A.: Security and Privacy in Cloud Computing: A Survey. Proceedings of the 6th International Conference on Semantics, Knowledge and Grids. pp. 105 112 (2010)

8 Empirical Research Fair

Empirical Research Fair Programme

Automation Supported Requirements Prioritization for Software Testing Purposes 185
Michael Felderer, Boban Celebic, Christian Haisjackl and Ruth Breu

Necessity of Electronic Requirements Negotiation 187
Georg Herzwurm, Mareike Schoop, Benedikt Krams and Sixten Schockert

Automation Supported Requirements Prioritization for Software Testing Purposes

Michael Felderer, Boban Celebic, Christian Haisjackl, Ruth Breu

University of Innsbruck, Technikerstr. 21a, 6020 Innsbruck, Austria
michael.felderer@uibk.ac.at

Problem Statement. Prioritization of requirements is a key element for decision support in all phases of requirements testing, i.e. test planning, design, execution and evaluation. The use of prioritized requirements to drive testing is a source of software cost savings. In current practice, prioritization is mainly performed by human experts based on aspects like importance or implementation cost applying various techniques like ranking or analytical hierarchy process. Therefore the process of requirements prioritization is often time-consuming and contains non-determinism. But especially for requirements prioritization in the context of testing also technical aspects like complexity of components or severity of identified failures are relevant. These technical aspects are based on artifacts like component design, source code or failure data and can therefore be measured automatically which is deterministic and time-saving. Thus, we develop an approach to requirements prioritization for the purpose of testing supported by automatically measured metrics of technical artifacts like the coupling or cohesion of Java classes and empirically evaluate its impact on the effectiveness and efficiency of requirements testing. As the automatic measurement of technical metrics is based on technical artifacts like source code or failure data, our automation supported prioritization approach can only be applied if artifacts like prototypes, similar software, or previous component versions are available. Especially for testing purposes this is often the case and makes our approach widely applicable. Additionally, in the context of testing, there is a demand for automated reprioritization based on technical artifacts in later test cycles. There are several empirical research challenges that we want to address by case studies in an industrial context. On the one hand we want to investigate the general conditions in terms of the types of technical metrics and the integration with classical requirements prioritization techniques under which the approach is beneficial. On the other hand we want to investigate the test effectiveness and efficiency of the approach. Test effectiveness addresses the degree of test goal achievement, and test efficiency addresses the resources consumption in order to achieve specific test goals.

Wanted from Industry. The requirements testing process implemented in the wanted industrial organization fulfills the following two prerequisites. (1) Requirements are prioritized based on an arbitrary classical requirements prioritization technique like ranking. (2) When prioritizing requirements, technical artifacts like prototypes, old components, similar software or previous versions of an existing or currently developed system are available and traceable to the requirements. It is not required that technical metrics are already measured automatically or that traceability links exist. We can automate the measurement and apply existing automated means to mine traceability links. Depending on the preference of the industrial organization, our approach can be directly integrated into the established testing process or be applied offline if the necessary artifacts are provided. For our approach we prefer software systems implemented based on the Java platform but are in principle not restricted to a specific technology or domain. The main benefit for the involved organization is a possible improvement of the requirements test process based on our empirical results.

Necessity of Electronic Requirements Negotiation

Georg Herzwurm[1], Mareike Schoop[2], Benedikt Krams[1], Sixten Schockert[1]

[1] Department for Business Administration and Information Systems II, esp. Business Software, University of Stuttgart, Keplerstr. 17, Stuttgart, Germany
{herzwurm|krams|schockert}@wius.bwi.uni-stuttgart.de
[2] Information Systems I,
University of Hohenheim, Schloss Osthof Nord, Stuttgart, Germany
m.schoop@uni-hohenheim.de

1 Goals and hypotheses

The goal of the proposed study is to investigate the impact of electronic support for requirements negotiations (RN) in the context of distributed software (SW) development. Collaborative support for RN becomes more and more important due to often asynchronous work of dislocated development teams. Stakeholders need to achieve a common understanding and need to agree on requirements ([6], p. 47) as conflict is at the bottom of each requirements engineering (RE) process. RN tools can provide dedicated support to these communication and decision finding processes.

Looking at software engineering research for the support of RE processes, several tools have been developed and are categorised in the so called SWEBOK ([1], p. 10-1); but none of the tools is dedicated to RN nor reflects the importance of communication as one way to avoid conflict and to reach consensus. Furthermore, basic office products are widely used in practice and more complex collaboration systems such as Microsoft SharePoint are adapted for RN, leading to lacks in usability, scalability, etc. due to missing methodological support of these tools.

Looking at negotiation research, electronic negotiations are not mere translations of traditional negotiations into the digital realm [7]. Rather, they provide additional value. Ströbel and Weinhardt define e-negotiation as being "(…) restricted by at least one rule that affects the decision-making or communication process, if this rule is enforced by the electronic medium supporting the negotiation, and if this support covers the execution of at least one decision-making or communication task." ([9], p. 147). Additionally, electronic negotiations should support document management [7]. Negotiation Support Systems (NSS) have been treated in the research area of RE but only focus on the negotiation of requirements itself or are not adopted onto the subject matter of RE properly.[1] Considering 'original' RN methods and tools it is noticeable that RN tools as EasyWinWin [1] have been further developed (ARENA, ARENA II, ARENA-M; [8]) but hardly reached market maturity. Likewise, the RN method DisIRE was implemented into a prototype but has not been mentioned since ([2], p. 206).

[1] Cf. [3], p. 154; example NSS are Inspire [4], SmartSettle [3], Negoisst [7].

We want to find answers to the question whether RN tools are useful in the context of distributed SW development which we strongly believe. Based on experiences with negotiations, we hypothesise that (1) *dedicated support through electronic RN enhances the quality of a RN process in distributed SW development*; (2) *dedicated support through electronic RN enhances the quality of the outcome of RN in distributed SW development.*

2 Description of the intended procedure and participants

To achieve valid answers to the above questions, an experiment will be conducted with expert participants who deal with distributed SW development in their profession. Ideally, the participants would have strong domain knowledge in RE and experiences in negotiation. Additionally, experiences in distributed SW development project(s) as well as practical knowledge in using SW engineering/ RE tools would be helpful.

In our experiment we will focus on one *case scenario*:

As part of the work of a distributed SW development team, elicitation of customers' requirements as well as the elicitation of product/quality functions have been performed. When it comes to the realization of product and quality functions, these need to get prioritized jointly by customers and developers depending on the function's ability to fulfill customers' requirements.

This is one point where conflicts typically arise and negotiations become necessary. Customers want all their high prioritized requirements to be realized, developers might only realize what is easily feasible within their product development strategy. Therefore, a conflict setting will be the starting point within the scenario for a test and a control group in the experiment. The test group will have electronic negotiation support and will use the electronic negotiation support system Negoisst [7] to solve the given negotiation task. The control group will not have electronic negotiation support and will be only allowed to use e-mail to communicate and to make offers or counter-offers to negotiate. Performance and quality of the RN process will be compared.

3 Benefits of the study and of performing the study

We believe that industry will benefit in these ways: As far as our investigations got so far, none of the 'big players' in RE tool business (e.g. IBM, Telelogic, Borland) offers a negotiation module within their SW up to now; only e.g. communication support for 'requirements discussions' are supported.[2] Against the background of the importance of distributed SW development, we think that our study might generate insights for the necessity of specific RN support in RE tools. Concerning the common reproach that RE tools are only worth for big companies due to their complexity [5], we believe that RN can lead to a monetary benefit for tool vendors as well as for SMEs in the long run. Furthermore, participants from industry have the possibility to reflect on

[2] E.g. Telelogic Doors 'Discussions': http://www.softqa.fi/pdf/doors_discussions.pdf.

their own approaches to distributed SW development by asking whether dedicated electronic support of RN might improve their SW development projects.

4 References

1. Abran, A., Moore, J.W.: Guide to the software engineering body of knowledge. Los Alamitos, IEEE Computer Society (2004)
2. Geisser, M., Heinzl, A., Hildenbrand, T., Rothlauf, F.: Verteiltes, internetbasiertes Requirements Engineering, in: WIRTSCHAFTSINFORMATIK 49(3), pp. 199 207 (2007)
3. Grünbacher, P., Seyff, N.: Requirements Negotiation, in: Aurum, A., Wohlin, C. (eds.): Engineering and managing software requirements, Heidelberg (2005)
4. Kersten, G.E.: Negotiation Support Systems and Negotiating Agents, in: Proceedings of EKAW'02, pp. 307 316 (1999)
5. Minne, S., Schockert, S.: Evaluierung von Requirements Management Tools anhand ausgewählter Anwendungsfälle, in: Stuttgarter Schriften zur Unternehmenssoftware Arbeitsbericht Nr. 3 (2010)
6. Pohl, K.: Requirements Engineering. Heidelberg (2010)
7. Schoop, M.: Support of Complex Electronic Negotiations, in: Kilgour, D.M., Eden, C. (eds.): Handbook of Group Decision and Negotiation, Heidelberg (2010)
8. Seyff, N., Hoyer, C., Kroiher, E., Grünbacher, P.: Enhancing GSS based Requirements Negotiation with Distributed and Mobile Tools, in: Proceedings of the 14th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05), (2005)
9. Ströbel, M., Weinhardt, C.: The Montreal Taxonomy for Electronic Negotiations, in: Group Decision and Negotiation (12), pp. 143-164 (2003)

Part III

REFSQ 2013 Doctoral Symposium Proceedings

9 Preface

Editor

Sergio España
PROS Research Centre, Universitat Politècnica de València, Spain, sergio.espana@pros.upv.es

Óscar Pastor
PROS Research Centre, Universitat Politècnica de València, Spain, opastor@pros.upv.es

Roel Wieringa
University of Twente, The Netherlands, roelw@cs.utwente.nl

REFSQ 2013 Doctoral Symposium foreword

Sergio España[1], Óscar Pastor[1], Roel Wieringa[2]

[1] PROS Research Centre, Universitat Politècnica de València, Spain
{sergio.espana, opastor} @pros.upv.es
[2] University of Twente, The Netherlands
roelw@cs.utwente.nl

Report from the Organizers

Conceiving and planning research is far from straightforward, but doing it is of utmost importance for all researchers whatever their area of interest. Researchers should not only identify open and relevant problems, they should also plan and conduct the research with rigour. Moreover, they must be able to communicate effectively so as to disseminate their findings.

Junior researchers should achieve such competences early in their careers, for which practice and constructive feedback are of great help. It is important that doctoral students expose themselves to the opinion of senior researchers and peers, even if it requires overcoming personal fears. That is the aim of a doctoral symposium. The REFSQ Doctoral Symposium offers the chance to PhD students in the area of requirements engineering to learn from expert researchers, report their research plans and results so far and receive feedback, share their experiences and expand their professional network.

In response to the call for papers, eight submissions were received, each including (i) a doctoral paper describing the context of their research, the problems observed and the goal of their PhD, the research questions refining the goal, the research methodology intended to address the questions, an outline of the proposed solution and a summary of the work performed so far; and (ii) a recommendation letter by one of their advisors advocating the relevance of the student's research for the requirements engineering community and overviewing the progress of the research project. Each submission was reviewed by one of the PC chairs and two PC members of the Doctoral Symposium and recommended for acceptance or rejection. Based on these evaluations, the chairs chose five doctoral papers to be presented in the symposium.

This year we run the third edition in Duisburg Essen. On the 8th of April, 2013, Roel Wieringa opened the REFSQ 2013 Doctoral Symposium welcoming all participants and outlining the programme. Then Sergio España conducted some warming-up exercises to help the students relax and, at the same time, energise all participants. Roel Wieringa introduced design science methodology to students, which provided a framework for discussion for the rest of the day.

Then each of the five students presented their research projects. Each presenter had a time slot of 50 minutes, with 20 minutes for presentation and 30 for discussion.

During their presentations, the PhD students in the room assessed the talk using anonymous evaluation forms. These were handed without further comments to the PhD students at the end of the day. During the discussion, one of the students in the audience recorded the minutes.

At the end of the day, there was a plenary discussion to reiterate lessons learned and discussion items that had come up during the day. Dan Berry offered his famous and hilarious talk on finishing a PhD on time. Bursts of laughter could be heard all over the venue but, more importantly, he provided useful guidelines and gave confidence to the students.

After the symposium, the presenters were invited to improve their doctoral papers, taking into consideration the advice and suggestions received. The reader can find these extended abstracts in the following pages.

We want to congratulate the students for their effort to deliver interesting presentations. We thank all the attendees for their participation in the discussions. Also, we acknowledge the REFSQ Doctoral Symposium Program Committee for their useful reviews. Finally we are grateful to the local organisers for their full support.

Doctoral Symposium Organization

10 Doctoral Symposium

Doctoral Symposium Programme

Developer-User Communication in Large-Scale IT Projects 199
Ulrike Abelein

Analysing the Assumed Benefits of Software Requirements 207
Richard Ellis-Braithwaite

Icon-based Language in the Context of Requirements Engineering 215
Sukanya Khanom

On Exploiting End-User Feedback in Requirements Engineering 223
Itzel Morales-Ramirez

A model-driven method to support conceptual model evolution 231
Marcela Ruiz

Developer-User Communication in Large-Scale IT Projects

Ulrike Abelein

Institute of Computer Science, University of Heidelberg, Im Neuenheimer Feld 326, 69120 Heidelberg, Germany
abelein@informatik.uni-heidelberg.de

Keywords: User-Developer Communication; User Participation, User Involvement, Software Intelligence

Abstract. User participation and involvement in software development are considered to be essential for a successful software system. In large-scale IT projects with traditional development methods the end user is mostly involved in the beginning of development (i.e. in the specification phase) and at the end of development (i.e. in the verification and validation phase). But there are also user-relevant important decisions in the phases in between (i.e. design and implementation). Thus, I argue that it is important to study how large- scale IT projects can enhance user-developer communication in order to increase system success. I investigated what evidence exists on effects of user participation and involvement on system success and explored which methods are available in literature and in practice through an interview series. In addition, the thesis will propose a method that supports large-scale IT projects in enhancing user-developer communication. As a first step I developed a descriptive classification containing user-relevant decisions and therefore trigger points to start user-developer communication. Furthermore a tool analysis and extension of one tool will ensure the feasibility of the method in real life large IT projects. Finally the feasibility and effects of the method will be evaluated in a case study.

1 Introduction

The complexity and scale of business software (SW) systems, such as enterprise resource planning (ERP) systems, has dramatically increased over the last decades [1]. Large-scale IT projects involve many stakeholders, whose different goals often lead to conflicting opinions and requirements. The resulting software system is supposed to be consistent with the desires of all stakeholders; therefore a need to involve stakeholder and in particular end users exists. In regard to user participation and involvement (UPI), there are three clear advantages of these large scale business system implementations. First, in comparison to new or evolutionary development of systems for a mass market, the prospective users are available within the company as are the developers who work long term on such systems. Second, a clear business

trend towards customized off-the-shell systems (COTS) over individual development of bespoke systems exists. As these standard systems already have best practices functionality built in, this leads to a defocus on actual functionality and a focus on customization. Thus, the technical challenges get less important, but the need to involve end users increases, due to their specific context knowledge. Third, in enterprises implementing these large-scale systems in comparison to mass market software, both end users and IT developer have an interest in achieving system success. The end users' work is dependent on the resulting system and IT personnel often have a hard position within large enterprises as their work is only perceived as support of the main business, thus they have an interest to improve their perception in the business domains.

So far most research on UPI focuses either on early or on late development phases [2], [3]. I believe that the step in software development when user requirements are refined (and thus interpreted) by developers into a technical specification (i.e. system requirements, architecture and models) is a critical one for UPI and specifically for user-developer communication. In this step a lot of implicit decisions are taken, some of which should be communicated to the end users. Thus, within the thesis, I plan to create a method that extends existing requirement engineering, software development and project management practice in order to enhance user-developer communication. Therefore, I identified trigger points (decisions that are made during software development) that initiate communication with the end users, developed a classification of user relevant decisions to define the granularity level on which to communicate with the end users, derived adequate means of communication based on the media richness theory, and will propose a setup enabling large-scale IT projects to enhance user-developer communication. To create the method I first did a synthesis of current research regarding studies of correlations between UPI and their effect on system success. Secondly, an analysis of existing methods for UPI in software development and IT project management helps to identify gaps to be closed with the new method. Third, an interview series has been conducted to validate a classification of user-relevant decisions in the design and implementation phases. Fours, the method will be detailed based on the interviews and validated in a case study to show the effects on system success.

2 Problems and Research Questions

Most large-scale IT projects are still using traditional project management and SW development methods like the waterfall model [4], [5]. Their advantages are high stability and clear agreements on price, timeline and scope [6]. However, the drawbacks are long periods of waiting for the business side [6]. Within these long development cycles requirements transform, as the translation from user to system requirements leads to a lot of interpretation and misunderstanding accompanied by a low level of user-developer communication. There are two effects: On one hand, end users do not feel integrated in the project. On the other hand, end users do not recognize their requirements in the acceptance phase [7]. Both effects lead to a low

acceptance of the system and a low motivation to participate in large-scale IT projects. In addition, a lot of frustration and inefficiency exists due to communication gaps between the project participants, in particular between the business users and the IT personnel (developers, designers and architects) [8]. Especially the backwards communication of decisions and their rationales after the requirement elicitation does mainly not exists in those projects [9]. A method that enables IT projects to enhance user-developer communication regarding rationale of decisions will help end-users to feel more integrated in the project and thus motivate them to support the project with their knowledge. This will not only lead to a higher system quality, but also a higher acceptance rate and usage of the resulting system.

Thus I want to answer the following questions in the thesis:

- RQ1- Does increased UPI lead to increased system success in large-scale IT project?
- RQ2 - What are the characteristics of existing methods in literature aiming to increase user participation and involvement in software development?
- RQ3 – How can a large scale IT project support user-developer communication (with a focus on the decisions and their rationale that are made in design and implementation phase) in order to increase system success?
- RQ4 – What effects has the method that supports large-scale IT projects in user-developer communication?

3 Proposed Solutions

3.1 Solution to RQ1 - Evidence that increased UPI increases system success

I conducted a systematic mapping study (submitted in December 2012 to the Journal for Empirical Software Engineering [10]), in which I identified empirical evidence in surveys and meta studies. I developed an overview of structural equation models that demonstrates that most papers showed positive correlations between aspects of development processes (incl. user participation), human aspects (incl. user involvement) and system success. Within the systematic mapping study, I extracted the researched aspects, correlation and number of participants for validation from 90 studies that were the result of our literature review. In order to analyze the aspects, I developed a classification with the main categories: development process, human aspects, system attributes, organizational factors and system success. The analysis revealed that *user participation and involvement is an important research topic,* as it has been researched in a broad manner by various research areas. The vast majority of the derived correlations showed a positive effect, thus I can conclude that *aspects of the development process and human aspects have a positive effect on system success***.** Another indicator for the wide range of this research area is the number of participants that were employed to validate the effects on a subcategory level. The analysis showed that *user participation and their involvement's positive effect on user satisfaction was validated by a more that 4000 participants involved in the surveys***.** Users, who feel involved, do use the system more frequently. Lastly, I looked into the

15 studies with negative correlations. Most of them show only a few negative correlations, but do not questions the main correlations between aspects of UPI and system success. In addition, I found out that *most studies with negative correlations were published a long time ago.*

3.2 Solution to RQ2 - Characteristics of methods increasing UPI

I looked into 27 methods papers within the systematic mapping study [10] and analyzed their targeted issue, their validation context and their proposed solution. I found out that *all software development activities (planning & project management, SW specification & requirement engineering, SW design & implementation, SW verification & validation, and SW evolution) are influenced by methods, but not many methods focus on the design and implementation activity*. The comparison between aspects researched by the surveys and meta studies and the targeted aspects from the methods reveals that methods for user participation and involvement target similar categories as the surveys and meta studies. But they do have a higher focus on the user-developer communication and the user's motivation. In addition, they target mostly the success factor system quality, which differs from the survey papers that mostly research user satisfaction. The analysis of the validation context revealed *that most methods were validated in a public environment.* The structured overview of practice with method examples shows *that practices derived from the solutions are distributed over all software activities.* In addition, I identified a focus on communication structures in the methods.

3.3 Solution to RQ3 – Support of user-developer communication

I will propose a method to support large-scale IT projects in enhancing user-developer communication with the four components: setup of communication structure based on stakeholder analysis, train developers on capturing decisions/changes, setup traceability of decisions, and define means of communication based on media richness theory. An interview series with twelve experts in large scale IT projects has been done. The results will be used to identify whether there is communication between end users and developers in large scale IT projects and if yes in what setup and phases it takes place. Furthermore, issues and consequences that are caused by communication gaps will be identified. I collected 81 examples of trigger points and thereof developed a descriptive classification for user-relevant decisions. The idea for a method has also been validated and will be improved and detailed..

3.4 Solution to RQ4 – Effects of method for user-developer communication

To assess the feasibility and effects of our method I will use the results of the interview series as well as the case study. I want to rate the usefulness of our method structure and hope to be able to measure improvements in user satisfaction when

testing the feasibility of the method within the case study. In addition, I will look into effects, such as is there an increase of direct user-developer communication and general communication interactions in the design and implementation phase after implementing the method?

3.5 Expected Benefits

Overall, the meta analysis showed a positive effect of UPI and in particular of user-developer communication on system success. Thus, a theoretical base of our method is available. The results give insights specifically for the community of human aspects in software engineering into the existing research on UPI. The overview of existing methods is useful for other researchers as they can see what method covers which targeted issues in which context and what the proposed solution is. In addition, it aids to understand the landscape of software development and IT project management methods in regard to UPI and indicates which parts of existing methods could be reused and combined in the new method. The descriptive classification of user-relevant decisions supports the method by helping developers and end users to understand important decisions and their implications. Through the interview series, I validated the classification and enhanced it with examples. The examples of decisions will help to explain, users, developers and researcher to understand when to start communication with the end users. The method will help to close communication gaps especially for the area of large-scale IT projects in a business context. There is a large bandwidth of existing methods, but a low usage rate within practice. Thus, it seems to be very hard to find the right balance between the developers' and end users' division of work and close alignment between these parties with a high level of communication. The method will describe in what situations it is useful to start communication with the end user (trigger points), how to structure that communication (when to inform whom on what granularity level) and how to represent these decisions and the rationale to help the end user to understand them. The validation of the method within a case study will also point to open issue and refinement needs within the method and rate the benefits.

4 Research Methods

The literature review is conducted as a systematic mapping study [11]. A search string has been used in twelve different sources from the domains (IT, Business and Communication). Overall 3136 hits have been identified, the initial selection based on publication title and abstract lead to a 232 downloaded publications. This selection has then been reviewed with clear exclusion criteria. The classification for end-user-relevant decisions is developed based on analysis of existing methods and then targeted for the context of large scale IT projects (RQ 3). The interview series was conducted with semi-structured interviews of twelve experts in large-scale IT projects. The evaluation of the method depends on the company for the case study. At

least one very large case study comprising a questionnaire of current situation and usage of user-developer communication, application of our method in (one or more) IT projects and analysis of implications on project success from the applied method will be done. Ideally this case study will be conducted by accompanying a real life IT implementation over a longer time period.

5 Related work

So far the topic of user participation in IT projects has mainly been researched in the information system field. This research mostly focused on the work-place context and looked for dependencies of UPI and system success. A broader approach has been taken by the research area of human-computer-interface. This area focuses on the design of interactive systems and their usability mostly known under the words of 'user-centered' or 'user-centered design' [2]. User-centered design utilizes methods such as task analysis, prototyping and usability evaluations [12]. Other forms are participatory design - focusing on democratic participation through workshops -, ethnography – emphasizing social aspects though observation- , and contextual design – looking into the context of work through contextual inquiry prototyping [13]. Within software engineering the topic has been of much interest, as neither user participation not user involvement is mentioned in the SWEBOK [2]. Despite this amount of existing research there are still gaps within the different methods and it is still an open question how user involvement should be integrated into SW development [14], [15]. Other methods such as participatory design based on the Scandinavian school, user-centered design defined in the ISO standard or joint-application-design [16] fail to point out how exactly (i.e. in which phases, which content, etc) the user involvement should take place [2]. So far most research focus on UPI either in the early development phases, e.g. requirement elicitation, or at the end of the development project within user acceptance tests [2], [3].An interesting study has been done by Bjarnason et al. (2011) [8]. They study communication gaps in terms of their (root) causes and effects (e.g. customer expectations that are not met, low motivation to contribute to the requirements work, software unit control of the implementation without alignment with the requirement team, unclear requirement coverage, quality issues and wasted effort from rework). Given those effects I believe that the step in development process when the user requirements are translated by the developer into the more technical specification of the system is a critical one. Even though most agile approaches implicitly use that sort of communication as they claim very close cooperation (mostly even physically together in one team room), the focus is more on a successful way to quickly develop working software. Besides that, these methods are hard to implement in large-scale IT projects. It is still an open question how the current high-ceremony methods can be extended by agile methods [5], as in most of the long term implementation projects the end users from business side cannot be a full time team member (as they need to perform their daily work).

6 Progress

I started the work on the PhD in September 2011. In the autumn of 2011, I have conducted the literature review and got familiar with the topic of user-developer communication. In 2012, I developed the first structure of the method based on TORE and the Media Richness Theory. Furthermore, I used the results of the systematic mapping study in order to answer RQ1 and RQ2. In addition, I conducted an interview series with twelve experts on large-scale IT projects. In 2013, I will analyze the results of the interview series and will detail the method. I will also include a tool analysis. Finally I want to conduct the case study for validation. I expect to finish the thesis by December 2013.

7 References

[1] S. Kanungo and S. Bagchi, "Understanding User Participation and Involvement in ERP Use," *Journal of Management Research*, vol. 1, no. 1, pp. 47–64, 2000.

[2] J. Iivari, H. Isomäki, and S. Pekkola, "The user - the great unknown of systems development: reasons, forms, challenges, experiences and intellectual contributions of user involvement," *Information Systems Journal*, vol. 20, no. 2, pp. 109–117, Mar. 2010.

[3] B. Ives and M. Olson, "User involvement and MIS success: a review of research," *Management science*, vol. 30, no. 5, pp. 586–603, 1984.

[4] R. D. Austin and R. L. Nolan, *How to manage ERP initiatives*. Boston: Division of Research, Harvard Business School, 1998.

[5] G. B. Alleman, "Agile project management methods for ERP : how to apply agile processes to complex COTS projects and live to tell about it," in *Extreme Programming and Agile Methods: XP/Agile Universe*, D. Wells and L. Williams, Eds. Springer Verlag, 2002, pp. 70–88.

[6] M. Fowler and J. Highsmith, "The agile manifesto," *Software Development*, vol. 9, no. August, pp. 28–35, 2001.

[7] W. J. Doll and G. Torkzadeh, "A discrepancy model of end-user computing involvement," *Management Science*, vol. 35, no. 10, pp. 1151–1171, 1989.

[8] E. Bjarnason, K. Wnuk, and B. B. Regnell, "Requirements are slipping through the gaps — A case study on causes & effects of communication gaps in large-scale software development," in *2011 IEEE 19th International Requirements Engineering Conference*, 2011, pp. 37–46.

[9] A. Al-Rawas, S. Easterbrook, U. S. N. Aeronautics, and S. Administration, "Communication problems in requirements engineering: a field study," *COGNITIVE SCIENCE RESEARCH PAPER-UNIVERSITY OF SUSSEX CSRP*, no. February, pp. 1–2, 1996.

[10] U. Abelein and B. Paech, "Understanding the Influence of User Participation and Involvement on System Success - a Systematic Mapping Study," *Journal of Empirical Software Engineering*, p. Submitted in Dec 2012, 2012.

[11] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," *Engineering*, vol. 2, no. EBSE 2007–001, 2007.

[12] M. A. Harris and H. R. Weistroffer, "A New Look at the Relationship between User Involvement in Systems Development and System Success Development and System Success," *Communications of the Association for Information Systems*, vol. 24, no. 1, pp. 739–756, 2009.

[13]S. Kujala, "User involvement : a review of the benefits and challenges," *Behaviour & Information Technology*, 2003.
[14]T. Lynch and S. Gregor, "User participation in decision support systems development: Influencing system outcomes," *Eur J Inf Syst*, vol. 13, no. 4, pp. 286–301, Nov. 2004.
[15]S. R. Humayoun, Y. Dubinsky, and T. Catarci, "A Three-Fold Integration Framework to Incorporate User – Centered Design into Agile Software Development," *Work*, vol. 6776, pp. 55–64, 2011.
[16]J. Wood and D. Silver, *Joint application development.* New York: John Wiley & Sons, 1995.

Analysing the Assumed Benefits of Software Requirements

Richard Ellis-Braithwaite
Loughborough University, United Kingdom
r.d.j.ellis braithwaite@lboro.ac.uk

Abstract. Often during the requirements engineering (RE) process, the value of a requirement is assessed, e.g., in requirement prioritisation, release planning, and trade off analysis. In order to support these activities, this research evaluates Goal Oriented Requirements Engineering (GORE) methods for the description of a requirement's value. Specifically, we investigate the goal to goal contribution relationship for its ability to demonstrate the value of a requirement, and propose that it is enriched with concepts such as correlation, confidence, and utility.

Keywords: software requirements, assumptions, benefits, strategic alignment

1 Problem Statement

There is often a "field-of-dreams" assumption that once software is built to the specified requirements, benefit will come [1]. The fact that there is little correlation between a company's level of IT investment and its profitability or market value, leading to the so-called "information paradox" [2], highlights the dangers of this assumption. Therefore, stakeholders responsible for a software project's funding need to be able to demonstrate that the software will be beneficial. Furthermore, practitioners performing RE processes where the benefit of a requirement is assessed (e.g., in prioritisation) need to know how benefit is defined by the stakeholders, and then how the requirements will contribute to it. With this in mind, this research aims to explore, improve, and evaluate methodologies for analysing the assumed benefits of requirements, and thereby, the alignment of those requirements to business strategy. Through evaluation of such methods in industrial projects, we aim to optimise the cost to benefit ratio of their application through methodology improvements, guidelines, and tool support.

The following research questions were formulated in response to problems faced by our industry partners, with the overall goal to improve decision-making in the RE process via better communication and analysis of assumed benefit:

- RQ1. What evidence exists to show that implemented requirements (i.e., software features and qualities) are not always beneficial?
- RQ2. What is an appropriate approach for modelling the assumed benefits of software requirements?
- RQ3. What aspects of the resulting benefit model are important for analysing the strategic alignment of software requirements?
- RQ4. What are the quality characteristics of such models, and what challenges preclude them?
- RQ5. How can a supporting tool address the challenges elicited from RQ4?

2 Motivation

The oft-cited, but ageing CHAOS report [3] suggests that two thirds of functional requirements which are specified before implementation are never or rarely used after their implementation. Such claims are supported by independent surveys, for example [4], which found that only 27% of the functionality in word processing software is ever used. Since stakeholders are often "motivate[d] to brainstorm requirements which they think that they just might need at some point" [5], it should be no surprise that some requirements lack pertinence, and as a result, deplete development resources.

Pertinence also affects the quality of non-functional requirements (NFR's). For example, a reliability requirement stipulating a certain level of service uptime should be the result of a trade-off made between two or more conflicting stakeholder goals, e.g., "maximize service availability" and "minimize infrastructure costs". Such trade-offs aim to maximise the utility of the software by optimising the associated cost-benefit ratios and acceptable risk levels [6]. Unfortunately, it has been found that stakeholders are rarely the source of NFR's, since "architects consider themselves to be the real experts when it comes to defining efficiency, reliability, and other similar aspects" [7]. On the contrary, the RE activity is primarily concerned with the description of the application domain [8] - the stakeholder's specialism, rather than the machine, which is the architect's specialism. In order to perform a successful trade-off, the rationale behind each goal is required, and without stakeholder involvement, over/under specification of NFR's will likely occur as a result of developer assumptions, ultimately leading to increased costs, delays, or in extreme cases, project failure [9]. Thus, for such consequences to be avoided, developers and stakeholders need to be able to comprehend the effects of a requirement's implementation on each other's goals, since the quality of any decision is underpinned by the information available to support it [10].

Numerous surveys blame the majority of software project failures, including poor return on investment (ROI), on inadequate RE [3, 11] - or more specifically, on poor stakeholder communication and incorrect assumptions [12]. Framing the problem in the context of these failure factors, we wish to minimise assumptions made about the benefits that stakeholders expect, by communicating those expectations to developers. Regardless of the cause, as a result of IT's poor ROI, IT-business strategy alignment has been the top ranking concern of business executives for the last two decades [13].

3 Related Work

The value based software engineering (VBSE) agenda [14] is motivated by the observation that most software projects fail because they don't deliver stakeholder value, yet, much software engineering practice is done in a value-neutral setting (e.g., where project cost and schedule is tracked rather than stakeholder value). Value-based requirements engineering (VBRE) takes the economic value of IT products into perspective through stakeholder identification, business case analysis, requirements prioritisation, and negotiation [15]. The primary VBRE methods are Business Case Analysis (BCA) and Benefits Realization Analysis (BRA) [16]. We consider other VBRE processes (e.g., prioritisation) as secondary, since they depend on benefit estimation.

In its simplest form, BCA involves calculating a system's ROI (financial benefits versus costs, in present value). An advancement from BCA, e^3value modelling seeks to understand the economic value of a system by mapping value exchanges between actors, ultimately leading to financial analysis such as discounted cash flow [17]. Howev-

er, such approaches are complex in their application, since the validity of any concise financial figure depends on assumptions holding true, e.g., that independent variables remain within expected ranges. Estimating benefit involves further intricacies such as uncertainty, and the translation of qualitative variables (e.g., software user happiness) to quantitative benefits (e.g., sales revenue) - none of which are made explicit by BCA.

BRA's fundamental concept is the Results Chain [18], which visually demonstrates traceability between an initiative (i.e., a software system) and its outcomes (i.e., benefits) using a directed graph, where nodes represent initiatives, outcomes, and assumptions, and edges represent contribution links. BRA's contribution links allow one initiative to spawn multiple outcomes, but the links are not quantitative, e.g., outcome: "reduced time to deliver product" can contribute to outcome: "increased sales" if assumption: "delivery time is an important buying criterion" holds true – but the quantitative relationship between "product delivery time" and "sales increase" is not explored. This poses a problem when outcomes are business objectives, since the satisfaction of an objective depends on the extent that it is contributed to, e.g., in the case of a cost reduction objective, the extent is the amount of reduction that will be contributed [19].

Goal Oriented Requirements Engineering (GORE) methods are capable of demonstrating alignment between software requirements and goals with AND/OR goal graphs [20]. Goal graphs ensure the pertinence of software requirements [20], since requirements must trace to a more abstract goal to explain the rationale for the requirement's implementation. Additionally, goal graphs improve communication between stakeholders since requirements are restated at various levels of abstraction (through goals), thereby bridging the gap between technically minded developers and application-domain focused stakeholders. Quantified goal graphs are used in [20] to model how far one goal contributes to its parent goals. However, the approach does not translate contribution scores toward various levels of goal abstraction, and therefore does not place the benefit contributed by a requirement into context, e.g., that a large saving may only be derived from a small cost [19]. A probabilistic layer for quantified goal graphs is proposed in [21] to reason about the probability that a goal's more abstract parent goals will be satisfied. However, this approach is time consuming, has limited applications, and does not capture stakeholder "attitude, preference and likings" [22].

Singh and Woo review the IT-business alignment literature within the RE field [23], and conclude that the majority of frameworks do not address business strategy or value analysis. The Strategic Alignment Model [24] is proposed as a theoretical framework for conceptualising IT-business alignment, but it is not taken beyond the conceptual level, and thus does not consider traceability to system requirements [25]. The OMG's standardised Business Motivation Model (BMM) [26] states that a set of quantified business objectives forms a business strategy, and that satisfaction of the objectives satisfies the strategy. Several methodologies have used the BMM to show the strategic alignment of software requirements. For example, in [27], OMG's SysML requirements metamodel is extended to establish traceability to the "tactic" concept from the BMM. Similarly, in [25], the BMM is used to decompose business strategy down to software requirements (i.e., from vision statements to tasks) which will satisfice the strategy. However, much like the BRA, neither approach takes a quantitative approach to demonstrating strategic alignment, yet strategic alignment depends on the extent of an objective's satisfaction, which is measured quantitatively.

4 Proposed Solution

The foundations for our approach to answering RQ2 are introduced in [19]. A brief summary is that the benefits of software requirements are stated at various levels of abstraction using goal graphs (where a benefit is the advantage gained by treating a problem and where goals are inverted problems). Goal graphs are used because they are well suited for visualising abstraction and refinement (hierarchies between parent/child goals), and can also visualise dependencies and cause-effect relationships [28].

A more detailed summary is that GRL goal graphs [29] are used to relate software requirements (represented by GRL task elements) to business objectives (represented by GRL hard goal elements) through GRL contribution links. Software requirements are represented by GRL tasks, in that it is the task of implementing the requirement that should satisfy some business objective. Business objectives are specified using the GQM+Strategies template [30], such that objectives can be considered as GRL hard goals (i.e., the objective's required magnitude is set). The proposed approach is: a) *prescriptive* in that some amount of business objective satisfaction is prescribed, b) *descriptive* in that the problem to be solved by the objectives is contextualised by various levels of goal abstraction, and c) *predictive* in that the contributions made by requirements or objectives to higher-level business objectives are quantitatively estimated.

In order to illustrate the approach's application, we refer to a software project that the authors were involved with. The software manages and schedules media files for digital signage (advertising). Fig. 1 shows an excerpt of a goal graph in the context of this project (in accordance with [19]), which explores some assumed benefits of a proposed functional requirement. A reader unfamiliar with GRL can find guidance in [29].

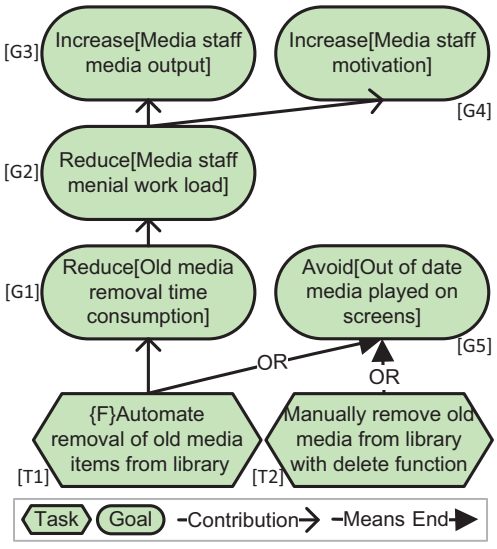

Fig. 1. Example goal graph showing the abstracted benefits of a proposed functional requirement

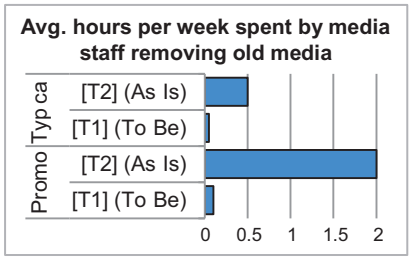

Fig. 2. Contribution of [T1] → [G1]

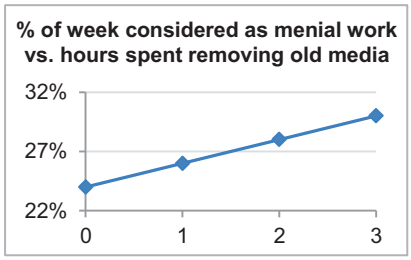

Fig. 3. Contribution of [G1]→[G2]

When it comes to answering RQ3, we are most interested in the contribution links between software requirements and business objectives, since they represent the alignment. Specifically, we are interested in describing *GoalX* to *GoalY* contribution

such that various levels of satisfaction of *GoalX* are mapped to various levels of satisfaction of *GoalY.* We are most interested in causal relationships between *GoalX* and *GoalY,* but in reality correlation and causation are difficult to distinguish in sociotechnical systems [31]. We also take into account the effects of the system's various usage profiles on goal contribution, as shown in Fig. 2 by the y-axis grouping (the "Promo" group represents a promotional period, such as a seasonal holiday).

For functional requirements, the contribution data is discrete, since there are only two states to capture - when the requirement is, or is not implemented. Thus, in Fig. 2, we refer to [T2] as the As-Is state to represent when the requirement is not implemented, and [T1] as the To-Be state to represent when the requirement is implemented. For non-functional requirements, the contribution data is continuous, since when properly specified, their values are numerical (e.g., an uptime requirement will have various possible states of satisfaction: 99.99%, 99.98%, etc.). Thus, if [T1] were non-functional, e.g., "the maximum time an expired media item should be displayed for is 5 minutes", then Fig. 2 would be an XY graph rather than a bar graph, with x-axis values specified in minutes and y-axis values derived from the goal that it contributes to. Note that soft goals are not used, since business objectives are defined quantitatively according to the BMM, and thus, qualitative contribution to quantitative goals is ambiguous.

Once the first contribution link has been described, we move up the goal graph to describe the next contribution link, as in Fig. 3, to provide context to the benefits (in Fig. 3, we show that the menial work is only partly comprised of old media removal). Enriching the contribution links with this information will allow "what-if?" analysis through interpolation and contribution propagation. Additional benefits include better elicitation of goal tolerance levels and improved clarification of the goal's criticality, i.e., the extent to which a goal's satisfaction criteria can be stretched (e.g., relaxing a non-functional requirement) without causing failure of the goals it that it contributes to.

To enrich the contribution link further, we make confidence explicit, that is, when a practitioner describes the correlation between two goals, their confidence in their descriptions will vary depending on their expertise and previous experience. Thus, capturing confidence (e.g., between 0 and 1) for each data point in a contribution link will allow decisions to be made in consideration of the assigner's uncertainty (lack of knowledge), previous accuracy in their confidence assignments, and their risk preference. Alternative approaches to representing confidence will be evaluated (e.g., enumerated estimate points {worst-case, likely and best-case}, intervals, and probability distributions).

Finally, root goals (those which do not contribute to other goals) are mapped to utility [10], where various levels of goal satisfaction result in various levels of "goodness". For example, referring to the root goal [G4] in Fig. 1, the various levels of staff motivation (e.g., measured on a likert scale between 0 and 5) would map to utility values (e.g., between 0 and 1). This will allow non-linear relationships between motivation and the utility of that motivation to be represented (e.g., that the difference between 0 and 1 on the motivation scale is bigger than 4 and 5). The concept of utility is both subjective and specific to the utility assigner. However, capturing it will explain the criticality of a root goal's satisfaction criteria, and any differences in utility assignment between stakeholders will be made apparent before the requirement is implemented. Thus, for this to be useful, the stakeholder's utility preferences must be communicated to each other for conflict resolution. Then, the stakeholder's utility functions can be aggregated in order to improve their integrity, as in the "wisdom of the crowd" theory [32].

RQ4 examines the qualities of a model used for the analysis of strategic alignment. The elicited qualities are much like the "completeness" quality of a requirements doc-

ument in that they are aspirational (their complete satisfaction is not expected). So far, we have elicited three core qualities from the application of our approach:

- **Determinism** – a goal graph produced by one practitioner relating a requirement to an organisation's goals should be similar to one produced by another. This can be supported by defining goals with metrics formalisation templates.
- **Transparency** – a goal graph should be self-explanatory rather than reliant on assumed widespread knowledge of what a goal means or why it matters. This can be achieved with goal abstraction to place goals into context.
- **Reusability –** the applicability of goal definitions (and their associated contribution links) to future projects should be as wide as possible. An example of poor reusability would be describing a contribution link relatively (such as "10% user task time reduction leads to 20% task cost reduction") without providing the absolute figures (e.g., *x* dollars were saved by reducing *y* hours).

The primary challenge to the first two qualities is that objective data is far rarer than subjective data. Decisions makers have found favour with inferior processes (e.g., qualitative goal contribution scoring) because they do not force you to think very hard [10]. For example, most managers will have the opinion that reducing menial work will improve employee motivation. Thus, assigning a "+" to that contribution link is easy, but understanding the extent to which menial work affects motivation requires a survey, since it might be the case that employees are unaffected by, or even enjoy menial work. Future work will attempt to understand which goal contribution links are in most need of analysis, since it is recognised that practitioner time is finite.

RQ5 looks at how far a tool can support the application of the proposed approach. A tool is currently under development[1], whose current features were derived from inefficiencies encountered whilst implementing the approach. For example, automatic goal graph drawing resulted from the observation that drawing large goal graphs is time consuming due to the number of edge collisions that occur. We plan to extend the tool's functionality with features such as goal similarity analysis (duplicate detection) in order to support the reuse of data from previous projects (e.g., benefits of a feature).

5 Research Method, Progress & Novelty

This PhD project is in collaboration with two industrial partners (LSC Group and Rolls-Royce)[2]. One industrial partner will provide case studies for software developed for their own organisation, while the other will provide case studies for software developed for an external organisation. The research approach adopted for this thesis is based on the experimental software engineering paradigm [33]. Firstly, a problem was identified with the help of our industrial partners. Structured interviews and questionnaires were then used to investigate the problem, which complements the motivation identified from the literature (briefly discussed in Section 2). Then, the scientific problem was defined in the format of research questions. After that, a solution idea was formed following a systematic literature review. A prototype tool was then developed to make the required data's capture, representation, and analysis possible so that the solution can be improved through feedback. The solution idea will then be validated against the scientific and the practical problem using case studies for evaluation.

---

[1] The prototype tool is available to download at http://www.goalviz.info/REFSQ DS/

[2] The author wishes to thank Dr. Tim King and Dr. Badr Haque for their support as industrial supervisors, and Dr. Russell Lock and Prof. Ray Dawson as academic supervisors.

This PhD project is half way through its three year duration, and an initial solution has been outlined. The remainder of the project will focus on case study research to evaluate the effectiveness of the approach compared to the current state of the art. Initial feedback from the tool's evaluation with practitioners is promising. Stakeholders, especially business managers, are attracted to the ability to understand technical software requirements in terms of the business objectives that they are familiar with. Feedback from developers has been more critical, since they are evaluated on the quality and timeliness of their programming, rather than on the value aspects of the software. Additionally, the numerical aspect of the approach has been off-putting to some.

As for novelty, we are not aware of an approach that considers the benefits of a software requirement as a chain of quantified goal abstractions. In particular, we are not aware of an approach that attempts to forecast the effect of a goal's satisfaction on its parent goal(s) at:

a) varying levels of goal satisfaction extent (explaining the effects of partial/full requirement satisfaction on multiple levels of goal abstraction);
b) varying levels of software usage (explaining the different profiles of software usage that can affect a requirement's contribution to goals);
c) varying levels of stakeholder confidence (explaining the extent to which a requirement's satisfaction may not contribute to a goal as specified);
d) varying levels of stakeholder utility (explaining the non-linear relationships between the extent of a goal's satisfaction and the utility gained);
e) varying levels of stakeholder agreement (explaining the variance between the stakeholder's estimates about the benefits that will be realised).

References

1. Boehm, B.: Requirements that handle IKIWISI, COTS, and rapid change. Computer. 33, 99–102 (2000).
2. Carr, N.G.: Does IT Matter? Information Technology and the Corrosion of Competitive Advantage. Harvard Business Review Press (2004).
3. CHAOS Chronicles v3.0. Standish Group International (2003).
4. McGrenere, J.: "Bloat": the objective and subject dimensions. CHI '00 Extended Abstracts on Human Factors in Computing Systems. pp. 337–338. ACM (2000).
5. Scott W. Ambler: Examining the "Big Requirements Up Front (BRUF) Approach," http://www.agilemodeling.com/essays/examiningBRUF.htm.
6. Jones, S., Wilikens, M., Morris, P., Masera, M.: Trust requirements in e-business. Commun. ACM. 43, 81–87 (2000).
7. Ameller, D., Ayala, C., Cabot, J., Franch, X.: How do software architects consider non-functional requirements: An exploratory study. Requirements Engineering Conference (RE), 20th IEEE International. pp. 41 –50 (2012).
8. Jackson, M.: Software requirements & specifications: a lexicon of practice, principles, and prejudices. ACM Press (1995).
9. Juristo, N., Moreno, A.M., Silva, A.: Is the European industry moving toward solving requirements engineering problems? IEEE Software. 19, 70 – 77 (2002).
10. Howard, R.A.: The foundations of decision analysis revisited. Advances in decision analysis: From foundations to applications. 32–56 (2007).
11. O. Vinter: From Problem Reports to Better Products. Improving Software Organizations: From Principles to Practice. Addison-Wesley (2001).

12. Gilb, T., Cockburn, A.: Point/Counterpoint. IEEE Software. 25, 64 –67 (2008).
13. J. Luftman, T. Ben-Zvi: Key Issues for IT Executives 2011: Cautious Optimism in Uncertain Economic Times. MIS Quarterly Executive.
14. Boehm, B.: Value-Based Software Engineering: Overview and Agenda. (2005).
15. Akkermans, J.M., Gordijn, J.: Value-based requirements engineering: exploring innovative e-commerce ideas. Requirements Engineering. 8, 114–134 (2003).
16. Boehm, B.: Value-based software engineering: Seven key elements and ethical considerations. Value-Based Software Engineering. 109–132 (2006).
17. Gordijn, J., Yu, E., van der Raadt, B.: E-service design using i* and e3value modeling. IEEE Software. 23, 26–33 (2006).
18. Thorp, J.: The Information Paradox: Realizing the Business Benefits of Information Technology. McGraw-Hill (1999).
19. Ellis-Braithwaite, R., Lock, R., Dawson, R., Haque, B.: Modelling the Strategic Alignment of Software Requirements using Goal Graphs. 7th International Conference on Software Engineering Advances. pp. 524–529 (2012).
20. Van Lamsweerde, A.: Reasoning about alternative requirements options. Conceptual Modeling: Foundations and Applications. 380–397 (2009).
21. Letier, E., Van Lamsweerde, A.: Reasoning about partial goal satisfaction for requirements and design engineering. ACM SIGSOFT SEN. pp. 53–62 (2004).
22. Liaskos, S., Jalman, R., Aranda, J.: On eliciting contribution measures in goal models. Requirements Engineering Conference (RE), 20th IEEE International. pp. 221 –230 (2012).
23. Singh, S., Woo, C.: Investigating business-IT alignment through multi-disciplinary goal concepts. Requirements Engineering. 14, 177–207 (2009).
24. Henderson, J.C., Venkatraman, N.: Understanding strategic alignment. Business Quarterly. 72–78 (1991).
25. Bleistein, S.J., Cox, K., Verner, J., Phalp, K.T.: B-SCP: A requirements analysis framework for validating strategic alignment of organizational IT based on strategy, context, and process. Information and Software Technology. 48, 846–868 (2006).
26. Object Management Group: BMM 1.1, http://www.omg.org/spec/BMM/1.1/.
27. Cui, X., Paige, R.: An Integrated Framework for System/Software Requirements Development Aligning with Business Motivations. 11th International Conference on Computer and Information Science. pp. 547 –552. IEEE (2012).
28. Van Lamsweerde, A.: Requirements engineering: from craft to discipline. 16th ACM SIGSOFT Foundations of Software Engineering. pp. 238–249 (2008).
29. Lin Liu: GRL Ontology, http://www.cs.toronto.edu/km/GRL/grl syntax.html.
30. Basili, V., Heidrich, J., Lindvall, M., Münch, J., Regardie, M., Rombach, D., Seaman, C., Trendowicz, A.: Bridging the gap between business strategy and software development. 28th International Conference on Information Systems, Montreal, Canada. pp. 1–16 (2007).
31. Sterman, J.: Business Dynamics: Systems Thinking and Modeling for a Complex World. McGraw-Hill/Irwin (2000).
32. Surowiecki, J.: The Wisdom of Crowds: Why the Many Are Smarter Than the Few. Abacus (2005).
33. V. R. Basili: The Experimental Paradigm in Software Engineering. Proc. of the International Workshop on ESE Issues: Critical Assessment and Future Directions. pp. 3–12. Springer London (1993).

Icon-based Language in the Context of Requirements Engineering

Sukanya Khanom

University of Jyväskylä, Department of Mathematical Information Technology
Jyväskylä, Finland

sukanya.s.khanom@student.jyu.fi

Abstract. Requirements engineering (RE) has been enormously intensified by the need for simple method, easy to learn, and the desire to communicate with stakeholders holding different backgrounds. This paper introduces the icon based language intended to describe constructs for expressing situations that take part during the requirements process. Icon based language utilizes features of visual notations standard in RE domain incorporating with icon representa tions. Icon based language is designed by combining meta modeling concepts and notations for functional and non functional requirements. The main appli cation area includes RE contexts such as elicitation, analysis, validation and traceability. The primary contribution is aimed at providing the stakeholders with an intuitive and convenient communication environment by using icon based language for describing wide range of applications from business goals and requirements narratives to high level system analysis and design.

Keywords: Requirements engineering, Icon based language, Meta model, Stakeholder.

1 Background and Motivation

Requirements engineering (RE) has exponentially become an essential part of software development process [1]. Several software development problems arise from shortcomings in terms that stakeholders elicit, document, agree and amend the software's requirements [1], [2]. To date, many mechanisms such as goal-oriented, UML and scenario devised to allow the development teams and other stakeholders to discover, specify and review requirements [3]. Unfortunately, one considerable deficiency is the fact that they require knowledge and skill to achieve the tasks. Resolving this fence will accelerate interaction and communication of all stakeholders. For that reason, the advantages of visualization [4], [5], [6] drive the research to make greatly improve and eliminate the host barriers of technical-rich methods such as misinterpretation, misunderstanding and misconception.

The aim of the current research is to introduce an uncomplicated visual modelling method which is based primarily on iconic counterparts. Visual representation is one of the main ways that human beings communicate: it is social practice [7], [8] varying

upon situations e.g. the sign language [8], diagrams [9], and comic illustrations [10]. The use of icons, symbols or signs in auxiliary communication makes visual representations different from natural language techniques that constructs on the basis of linear orderings of words [11]. Icons' meaning can be perceived straightly and they also encourage communication across international frontiers. Astonishingly, icons have been accepted successfully in human-computer interface, but seldom in RE visual notations [12], [13].

In this paper, we propose an icon-based language as a communication means for different stakeholders in RE. The study is focused to the construction of feasible visual sentences. The visual sentence is the composition of visual vocabulary, syntax, and semantics. Each construction is understood as a representation of a concept, an object, an action, or a relation.

The rest of the paper is structured as follows. The next section starts with the related work. Then, the research questions and methodology is presented. The following section presents the proposed solution of icon-based language in RE process. After that, up-to-date progress is explained. And final section is reserved for conclusion.

2 State of the Art

Moody [13] indicates that icons and visuals represent important benefit for communication research, especially to communicate about topics in which sound difficult for novices. Research attempts have been made to develop computer-intensive iconic communication systems, which primary aimed at fostering people to communicate with each other. Some systems are dedicated as a communication tool for the people who have speech disorders [8], [14]. Several proposals are rudimentary on linguistic theories such as the conceptual dependency theory introducing the concepts that the units of meaning correspond to the grammatical units of clause and words (e.g. [11], [15]). A profound research has been completed on designing the system that facilitates the reviewers to mutually communicate without sharing common language [15]. In the field of crisis management, Fitrianie et al. [11] announced a comprehensive icon-based interface that exploits graphic symbols to represent concepts or ideas.

In the RE community, wide variety of researches has highlighted on diagram for improving requirements engineering activities [16], [17], [18]. Extended features of use case diagram have been devised by Yang-Turner et al. [16] to support the tasks of stakeholders in elicitation activity. In similar manner, Helming et al. [18] approached an incremental UML as a communication means for delivering collaborative environment whereas, Cardei et al. [17] adapted the UML methodology into specification and validation phases to alleviate the gaps of requirements ambiguities and misinterpretations. Most diagrams make very diminutive use of semantic transparency and are abstract shapes whose meaning are articulately conventional and must be learnt. Visual notation in software engineering has been studies extensively by Moody [13] to define a series of principles for designing cognitively effective visual notation. He also advised to use pictorial icons that their meaning can be conceived directly and easily learnt to enhance cognitive effectiveness in all stages of RE process.

3 Research Questions

The premise research questions (RQ) of this paper are as the following:

- RQ 1: What are the difficulties users currently experienced in performing RE?
  - RQ 1-1: What are the existing problems still left to be solved?
- RQ 2: How can we support the tasks of software developers and other stakeholders in RE process with icon-based language, especially in multi-cultural environments?
  - RQ 2-1: What are the tasks of RE that can be supported by icons, for instance, requirements attributes and RE process etc.?
  - RQ 2-2: How can the concept be designed to take into account of the different cultures and behaviors in effective RE?
  - RQ 2-3: Who are the key stakeholders to be benefited from a proposed solution?
- RQ 3: How to validate if a proposed solution supports RE stakeholders and is easy to learn and understand?

In order to answer those questions, we use the design science research for icon-based language development as depicted in Fig. 1:

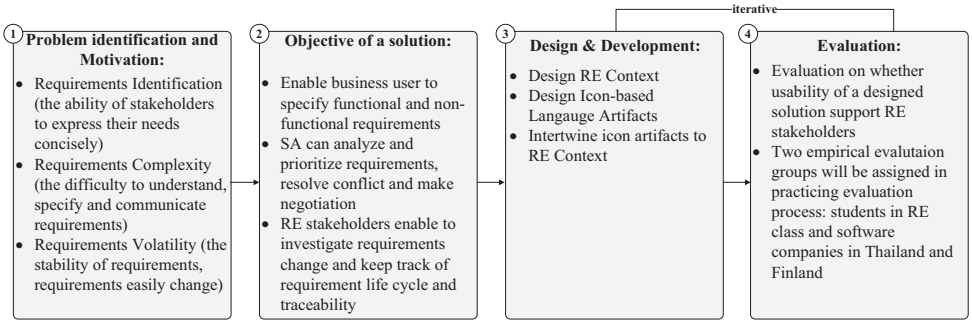

Fig. 1. Design science research methodology for icon based language

We first identify the problem of the whole RE context to understand real interests of stakeholders. According to literature review, we arrived at three problem identification in performing RE [19], [20]. Firstly, there is the ability challenge of system stakeholders to express their needs concisely and concretely. In broad spectrum, requirements are heavily hard to discover in situations where there is a communication gap between technical and non-technical users that appear to speak different languages and apply different approaches for desired outcomes. Secondly, requirements complexity happens when stakeholders encounter the difficulty to understand, specify and communicate requirements. Finally, requirements volatility signifies to the stability of requirements that easily change as a result of environmental dynamic or individual learning. We then define concrete objectives to inform the necessities of a possible solution to the aforementioned problems. Our main objective is to find a solution that provisions the RE stakeholders who have been influenced by multicul-

tural backgrounds to specify requirements, analyze and prioritize requirements, resolve conflicts and make negotiation, and examine requirements change, requirements life cycle and traceability of requirements. At the design stage (see number 3 in Fig. 1), we will develop RE context and icon artifacts including the integration of those two worlds by means of Re process which begins with patterning the scope and vision, use case scenario and ends up with requirements specification. Following the theoretical evaluation, we empirically assess the utility and usability [21] of icon-based language whether it is simple enough to comprehend by RE stakeholders.

4 Proposed Solution: Icon-based Language

Icon-based language is proposed to provide RE stakeholders with technique that does not require advance knowledge, the development of more lightweight and intuitive interaction. In RE context, it is important to take into consideration the life cycle of the requirements as well as the attributes of the requirements. Icon-based language is defined rooted on the visual vocabularies visual grammars (syntax) and semantics [13], [15], [22].

- **Visual vocabularies or graphic symbols:** visual vocabulary is set of icons that annotate the visual sentence schemed on one-dimensional, two-dimensional and three-dimensional space and possibly associated through special relationships (see an example in Fig. 2). The iconic symbols consist of four actor types, three node types, five priority types, five status states, three dependency relationships, two parent-child relationships, two link types and one measurement bar for number of changes. Overall, actor represents stakeholders and systems that have the purpose and actions to achieve goals. Node symbolizes the various types of RE activities such as requirements taxonomy or elicitation tasks that we categorize to be individually exemplified by icon(s). For example, "goal" represents business requirements. Rationale signifies the scenario of activity that will be presented by icon(s). Typically, rationale dynamically describes the requirements' behavior under various conditions such as a group "priority" and a series of "status" states.

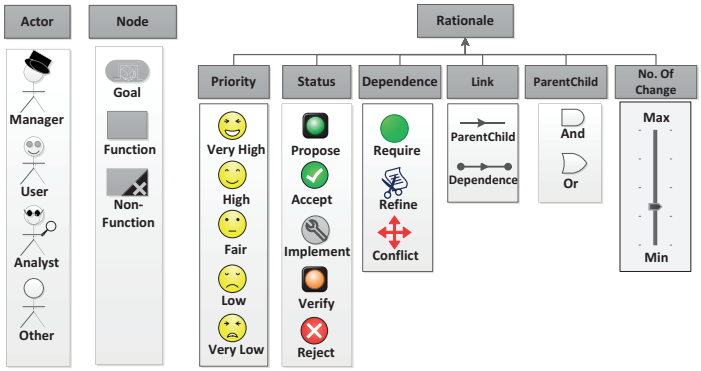

Fig. 2. Icon vocabularies for icon based language corresponding to a semantic structure

- **Visual grammatical rules/Syntaxes:** A set of icon elements and visual grammatical rules are altogether compounded to visual syntax. Currently, visual syntax is designed by applying the "Physics of Notation" theory [13]. We classify the visual vocabularies into three categories: actor, node and rationale as shown in Fig 2. Stick figures would be used to represent actors because they are universally interpreted for the representation of people. Variations of stick figures could help reader to distinguish the different types of actors. For example a stick wearing hat can be representative of manager. Node elements bear a resemblance to concrete icons together with geometrical shapes that are, however, globally accepted. For the remainder, we use abstract objects that are easily recognized such as the mathematical signs emotional faces, different line connectors of arrows and logical signs. In addition, color is used to improve cognitive effectiveness such as green, red and yellow colors which are the standard color for traffic sign.
- **Visual semantics:** The semantics of an icon-based language is represented in the meta-model form. Each syntactic creature is arranged to some semantic construct. As an example of parent-child relationship described in Fig. 3, the announcement of two banking transactions for account inquiry and money transfer involves several relevant requirements. Some of them, for instance, are:
  - FunReq 001: The system shall provide customer two transaction types, account inquiry and money transfer.
  - FunReq 001 1: Customer is able to inquiry only accounts that have been added into the online system.
  - FunReq 001 2: There are three categories of money transfer, (1) 1st party fund transfer: transferring within the same bank, different account but same owner name, (2) 3rd party fund transfer: transferring within the same bank and different owner account, and (3), other bank fund transfer: transferring over different bank account.
  - FunReq 001 3: List of accounts shall be displayed in alphabet order.

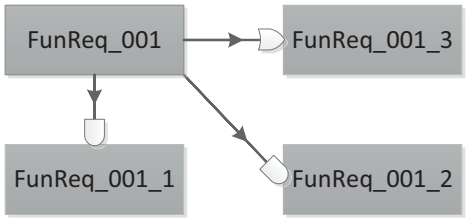

Fig. 3. An example of parent child relationships and semantics

The meaning of an example in Fig. 3 can be inferred as to justify a parent requirement (FunReq 001), it requires two children requirements of FunReq 001 1 and FunReq 001 2 to be justified, too whereas a parent requirement (FunReq 001) can be either fulfilled or not fulfilled by a child requirement (FunReq 001 3).

4.1 Icon-based Language Meta-model

The icon-based language meta-model (ILModelElement) demonstrated in Fig. 4 has been defined on the industry standard conventions (e.g. [23], [24]). In order to model general characteristics, the core meta-model includes entities such as Actor, Relationship, and Requirements. This meta-model can describe and communicate requirements as well as structure the reasoning about them. Actor definitions are frequently used to represent stakeholders or systems. Requirement elements are a linkable element to give an account of and the reasons for the proposed require goals and desires. A Requirement is characteristics of requirements artifacts that have a unique identifier (ID property), a name, a description, a priority (priority type), and a status (status type). A model may contain additional requirements (AdditonalReq) which is customizable. Requirement can have relation with each other by three dependency types: Refine, Require and Conflict. A number of changes is enclosed to a corresponding requirement to provide extra information about the requirements volatility.

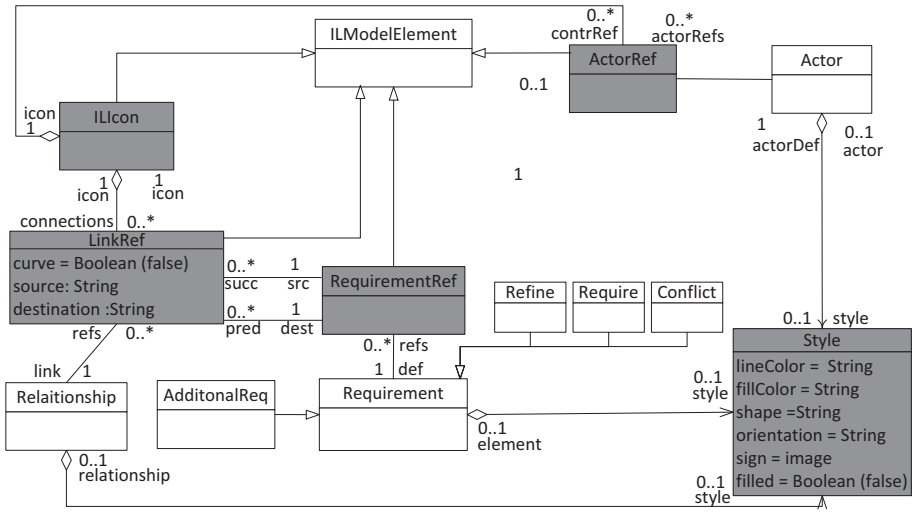

Fig. 4. Meta model for icon based language

Icon-based language-Icon (ILIcon) is a container for all actor reference, requirement reference, and link reference. An actor reference (ActorRef) refers to an actor definition and show its boundary. A link reference (LinkRef) is a direct link that bonds a source element to a different target element. A requirement reference (RequirementRef) shows a requirement element. Its representation associates with the type of the requirement element definition it refers to. The syntax such as color shape, orientation or symbol of an actor, relationship and requirement definition are defined in concrete style (Style) and are therefore shared by all the actors, requirements and relationship types. Ultimately, the concept of core meta-class of icon-based language is flexible for further customization.

5 Research Progress

As part of development phase of the research project, an initial attempt to produce an icon-based language, using RE contexts as a theme is skeletonized on the conclusion of interdisciplinary literature review (phase 1). Key attributes of icon-based language have been implied: - that RE world are identified, - that icon artifacts are designed to support the defined RE world and - that every construct in the language is represented visually by iconic symbols.

In the near future, the empirical evolution needs to be set up for testing the icons. It includes the formulation of questionnaire and test case scenarios. Different kinds of questions will be asked when testing icons: open ended (e.g. having icons available and ask what fitness describes the meaning of those icons?), image meaning (i.e. giving permission to the subject to match icons and meanings from two lists: one of icons and another one of meanings), and icon category (e.g. arranging a specific set of icons and asks the subject which category does those icons belong to?). Data is collected using a combination of methods. Key components are the user test and the user satisfaction questionnaire.

During empirical evaluation, we conduct two iterations: one with multicultural students in the RE course of the Department of Mathematical Information Technology at the University of Jyväskylä, and another with software companies both in Thailand and Finland. The best way to reach the heterogeneous participants is web-based icon test. By having the survey dispatched on the Internet, it can possibly grasp any person in any location that has access to the Internet. We will form an electronic survey and put it on the web. The result of these two iterations will be used to measure if the icon-based language is easy learnt and inform improvement possibility.

6 Conclusion

RE has been widely adapted by various communities while it remains a huge challenge in the context of interoperability between stakeholders. This research intends to propose an important framework encompassed with icon protocol towards breaking down communication obstacle between development teams and business stakeholders. We define a number of RE contexts that are needed to express requirements and to be represented by icons. The main expectation constitutes to human-oriented perspective that icon-based language is flexible and simple enough to allow stakeholders to apprehend. Icon-based language is designed rooted on the concept of visual notation and iconic communication that goes beyond the traditional natural languages. Throughout the research, it will contribute significantly to some possible ways for solving the problems caused by language and technical impediments when delivering requirements in software development life cycle. Furthermore, a simple restricted grammar and self-explanatory icons would make the icon-based language more appealing. The icons used for icon-based language should be commonly recognized across cultures. Otherwise, the icons might be designed for localization special icons for each of target cultures.

References

1. Pohl K.: Requirements Engineering Fundamentals Principles and Techniques. Springer, Berlin (2010)
2. Wiegers K E.: Software Requirements, Second Edition, Microsoft Press (2003)
3. Agarwal B. B., Tayal S. P., Gupta M.: Software Engineering & Testing (Computer sci ence series). In:Jones and Bartlett Publishers (2010)
4. Khanom S., Heimbürger A., Kärkkäinen T.: Icon based Language in Requirements Devel opment. In: Information Modelling and Knowledge Bases XXIII. IOS Press (2013)
5. Heimburger A, Kiyoki Y., Ylikotila T.: Communication Across Cultures in the Context of Multicultural, Software Development. In: Reports of MIT. Series C. (2011)
6. Heimburger A., Kiyoki Y., Kohtala S.: Intelligent Icons for Cross Cultural Knowledge Searching. 21st EJC on Information Modelling and Knowledge Bases, June (2011)
7. Tolar T. D., Lederberge A. R., Gokhale S., Tomasello M.: The Development of the Ability to Recognize the Meaning of Iconic Signs. Journal of Deaf Studies and Deaf Edu., (2008)
8. Choo K., Woo Y., Min H., Jo J.: Icon Language Based Auxiliary Communication System Interface for Language Disorders. Advances in Multimedia Information Systems (2005)
9. Moody D. L.: The "Physics" of Notations: Towards a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Tran. on Software Engineering, vol. 35, no. 5, pp. 756 778, Dec (2009)
10. Cohn N.: The Visual Language Manifesto Restricting the "Comics" Industry and Its Ideol ogy. In:Second Edition, Emaki Productions (2007)
11. Fitrianie S., Datcu D., Rothkrantz L. J.M.: Human Communication Based on Icons in Cri sis Environments. Usability and International. Global and Local User Interface (2007)
12. Chang S. K., Polese G., Orefice S., Tucci M.: A Methodology and Interactive Environment for Icon based Language Design. In:Journal of Human Computer Studies (1994)
13. Moody D. L., Heymans P., Matulevicius R.: Visual syntax does matter: improving the cognitive effectiveness of the i* visual notation. Requirements Engineering, Springer Verlag, London Limited (2010)
14. Basu A., et al..: Vernacular Education and Communication Tool for the People with Mul tiple Disabilities. In:Development by Design Conference. Bangalore (2002)
15. Leemans N. E. M.: VIL: A Visual Inter Lingua. PhD thesis (2001)
16. Yang Turner F., Lau L.: Extending Use Case Diagrams to Support Requirements Discov ery. Workshop on RE for Systems, Services and Systems of Systems (RESS) (2011)
17. Cardei I., Fonoage M., Shankar R.: Model Based Requirements Specification and Valida tion for Component Architectures. 2nd Annual IEEE Systems Conference (2008)
18. Helming J., et al.: Towards a Unified Requirements Modeling Language. In:IEEE 5th In ternational Workshop on Requirements Engineering Visualization (REV) (2010)
19. Hansen S., and Lyytinen, K. L.: Challenges in Contemporary Requirement Practice, IEEE Proceedings of the 43rd Hawaii International Conference on System Sciences, 2010.
20. Mathiassen L., Tuunanen T., Saarinen T. and Rossi M.: A Contigency Model for Require ments Development, JAIS, Vol. 8, pp. 569 597, 2007.
21. Galitz W. O.: The Essential Guide to User Interface Design: An Introduction to GUI De sign Principles and Techniques. In:Third Edition, Wiley Publishing, Inc. (2007)
22. Harel D., Rumpe B.: Meaningful Modeling: What's the Semantics of "Semantics"?. In:IEEE Computer Society, pp. 64 72 (2004)
23. ISO/IEC, ISO/lEC Standard 24744.: Software Engineering Metamodel for Development Methodologies. In: ISO/IEC (2007)
24. ITU T.: User Requirements Notation (URN). Z.151, Telecommunication Sector (2008)

On Exploiting End-User Feedback in Requirements Engineering

Itzel Morales-Ramirez

Center for Information and Communication Technology, FBK-ICT
International Doctoral School ICT- University of Trento, Italy
imramirez@fbk.eu

1 Motivation and Problem

Requirements elicitation is a relevant research area in Requirements Engineering (RE) since elicitation is a key task that must be performed in any project [12]. Elicitation usually involves different types of stakeholders (e.g. analysts, managers, users, etc.) who provide information that may turn into requirements. Over time, several elicitation techniques have been proposed to gather requirements. But how and when end-users express their expectations about a software is an issue that has been mainly explored within the user experience research community [7], and it is only recently that RE is looking at it [22]. User feedback is recognised as a source for changes in a system [6] (i.e. changes in requirements, new ones, etc.).

We define *end-user feedback* as *meaningful information provided by end-users of widely used software applications with the purpose of suggesting improvements to such applications (i.e. new needs, modifications, or strategic behaviours)*. End-user feedback in discussion forums seems to be helpful for developers in evolving applications. Specifically, the continuous management of fansites (e.g. blogs, discussion forums, etc.) of an online game (i.e. Habbo Hotel[1]) let developers discover the emerging needs of end-users in a participatory design approach, as remarked in [11]. The perspective about feedback proposed by Jo et al. [13] is from the viewpoint of reporting software defects. The open bug reporting of the Mozilla project is the case study of this research. They investigate how users (e.g. core developers, power users, etc.) contribute in reporting problems. The results show that the open bug reporting experience is not fully successful in obtaining valuable feedback but in recruiting talented developers.

Therefore, we have identified the following main problems when managing end-user feedback: i) the heterogeneity of abstraction levels in which it is written, ii) the huge amount of it, and iii) the mismatching of its purpose, which is wrongly classified as a bug. Based on this, we believe that a more structured end-user feedback could be a promising support towards an effective management and analysis in RE, thus obtaining meaningful information. The establishment of such a structure, hence, can be useful for analysts in discovering requirements

[1] http://www.habbo.com

knowledge. Nevertheless, users might not feel fully stimulated in providing feedback whether they are asked to follow a certain structure. An attempt to address this issue is presented in [3], for instance, this work provides a document structure for organising requirements and reports on wikis easy to learn and useful for stakeholder collaboration.

As example of the current practice in gathering feedback there are commercial applications (e.g. IdeaScale [9], UserVoice [10]) that help enterprises in obtaining feedback from customers or stakeholders. But their focus is mainly based on a voting mechanism and feedback is likely to contain redundant ideas. Other works have proposed the acquisition of requirements through the use of mobile technology. Seyff et al. [23] have developed a mobile requirements engineering tool, iRequire, that allows users to report needs. The work in [22] presents the ConTexter tool that supports the collection of spontaneous feedback. The gathering of feedback is guided by a list of entities that are sensed by using GPS or Bluetooth. Our proposal differs from these works since we want to apply a structured discussion and we do not limit collecting feedback only through mobile devices.

The work by Qureshi et al. [21] proposes the Continuous Adaptive Requirements Engineering framework that continuously captures changes of end-users' requirements. The ultimate goal is the development of a tool for an online RE method with the aim of enabling system self-adaptation, but it is not explained how the needs must be identified or requested. Unlike them, our proposal intends to identify and characterise end-user feedback as a source of new needs, modifications, etc., as a preliminary step towards defining techniques for an effective gathering.

On the other hand, a research work that applies human computer interaction (HCI) principles and methods in an RE process is the work by Sutcliffe et al. [24]. They highlight the fact that HCI and RE share the problem of interaction with users to get their requirements. One of the objectives of their work is the exploration of HCI techniques in an RE process. Some of the techniques they use are: paper prototypes and user feedback to evaluate it, however, we think that user feedback is not fully exploited in their approach.

Based on the previous observations we formulate our research objective (RO) in terms of the following design problem: *Define a systematic approach for acquiring end-user feedback and deriving requirements knowledge from it.* Concrete research questions (RQ) are presented below following a design science approach [25], i.e. knowledge question (KQ) and practical problem (PP):

RQ1 (KQ) What are the current interpretations of end-user feedback, found in the literature, that can serve to its conceptualisation for the purpose of improving its management in RE?

RQ1.1 (KQ) What are the concepts that compose end-user feedback?

RQ1.2 (KQ) What are the relationships among these concepts?

It is important to define what are the conceptual entities of feedback that will help analysts in classifying and identifying new system behaviours (i.e. functions and qualities), thus motivating the system evolution.

RQ2 (PP) Which are the promising candidate elicitation techniques to collect explicit[2], direct and indirect, end-user feedback?

RQ2.1 (KQ) What are the current problems when collecting end-user feedback, according to the literature?

RQ2.2 (PP) Which techniques can improve the collection of direct feedback?

RQ2.3 (PP) Which techniques can improve the collection of indirect feedback?

These questions are addressed along with the support of other research areas such as user engagement and social computing. By taking advantage of ideas and techniques of these areas for engaging users to provide feedback.

RQ3 (PP) How can analysts derive requirements knowledge from the collected end-user feedback?

RQ3.1 (PP) What are the terms contained in this feedback that might be extracted?

RQ3.2 (PP) What are the characteristics of the terms that must be considered for mapping these terms to existing requirements specifications?

These questions refer to the discovery of candidate requirements or identified changes in existing specifications.

RQ4 (PP) What are the validations that must be elaborated to assess if the proposed approach improves the management of end-user feedback, for deriving requirements knowledge?

RQ4.1 (PP) Do the concepts composing end-user feedback interpret what we intend to study?

RQ4.2 (PP) Do the concepts define what is observed in real feedback?

RQ4.3 (PP) Is the conceptualisation of end-user feedback useful for supporting the analysis of feedback expressed in natural language?

The rest of the paper is structured as follows. In Section 2 is described the approach we propose to address the previous questions. The research methodology, plan and validation are presented in Section 3 and the progress is given in Section 4.

2 Proposed Approach

We aim to adopt a multidisciplinary perspective to address the formulated questions. In Figure 1 we present the elements exploited by our proposed approach. In the rectangular shapes candidate techniques are highlighted indicating their research area (first line) and a number, which correspond to the framework component we intend to exploit for. For instance, we want to take advantages of RE techniques for the elicitation of feedback, number 1, by structuring the gathering of it based on a topic formulated on key terms expressed in the system's behaviour. In this example the behaviour is represented as a goal-oriented modelling. Another example, rectangle number 3 refers to the exploitation of some natural language (NL) techniques that might support the extraction of such key

[2] Explained in Section2.1

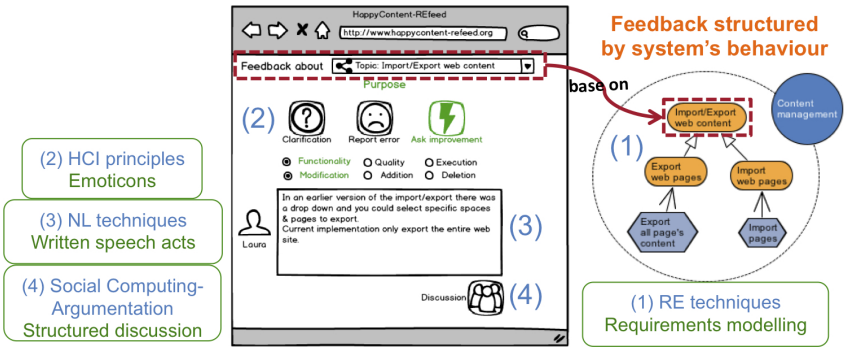

Fig. 1. Elements exploited by the proposed approach.

terms from end-user feedback. However, to achieve an analysis of feedback, we first need to understand and clarify the concepts composing feedback, as stated in our *RQ1*. In the following subsections we briefly present preliminary ideas of the approach according to the stated research questions.

2.1 Conceptualisation of End-User Feedback

Next, we explain a characterisation of feedback based on what we have found in the literature. The *acquisition* of feedback can be *explicit* or *active* and *implicit* or *passive*. The former is obtained through a direct indication in providing it. While, *implicit* feedback is collected on monitoring behaviour in an unobtrusive manner [1, 14]. The *communication* of feedback between actors can be *direct* when a user communicates a concrete need to analysts team or she/he sends a log of information to them, and *indirect*, when the feedback is shared and discussed within a community of users [8]. According to the *purpose*, the reason for which feedback is created, can be *corrective* or *negative*, *encouraging* or *positive*, to promote a *strategic behaviour*, and to provide additional *clarification*. In [2] the *negative feedback* is a type of feedback in nature that tries to neutralise a perturbation. On the other hand, *corrective feedback* provides information about how well a task is being performed [8]. The opposite types are *encouraging* or *positive*. The former is defined as a motivating manner for indicating the directions that a student could pursue. The latter reinforces a perturbation in systems in nature and leads to an amplification of that perturbation [2]. The *strategic behaviour* helps in giving several options for achieving a certain process [20]. The feedback can be a *clarification* when it contains extra information, such as critical details, that makes goals clearer [20]. An excerpt of such a characterisation, showing the purpose, is presented in Table 1. The first column refers to the dimensions of our characterisation (label *Dimension*), the second to the different types of classification of feedback we identified for each dimension (label *Classification*), and the third column reports an example for a given dimension-classification value.

Table 1. Excerpt of the characterisation of types of feedback.

Dimension	Classification	Example
Purpose	-Corrective (Negative)	When the human body experiences a high concentration of blood sugar, it releases insulin, resulting in glucose absorption, and bringing the blood sugar back to the normal concentration.
	-Encouraging (Positive)	Ants lay down a pheromone that attracts other ants. When an ant travels down a path and finds food, the pheromone attracts other ants to the path.
	-Strategic behaviour	A peer can give another strategy for achieving a specific activity.
	-Clarification	A book can provide information to clarify ideas.

2.2 Collecting End-User Feedback

As previously mentioned, we will take advantage of other research areas to boost the proposed elicitation technique to collect feedback. From HCI we intend to borrow techniques for engaging users[3], see number 2 in Figure 1, because users feel more motivated in providing feedback when they perceive their ideas are being truly considered [4]. From social computing we are considering to exploit an online discussion platform to enable gathering structured feedback. Since the gathered feedback will be used for improving a software system, this means that end-users must discuss about the improvements suggested and somehow validate the diverse opinions. To perform such a discussion we believe that an argumentation-based approach will be appropriate to guide a structured discussion, number 4 in Figure 1. We are considering to adapt the work in [12], which describes the framework ACE (Acceptability Evaluation) that focuses on discussions with the purpose of concluding the validity of an RE artefact. On the other side, for the engagement in collecting end-user feedback we are considering to look at state-of-the-art works that present results in the use of emotion, colours and words. Emotions and feelings are not only limited to human-interaction and are also present while interacting with software systems. A benefit of using colours can be to convey a message quickly and in an easily understandable manner [16].

2.3 Discovering Requirements Knowledge

This part of our research is focused on techniques to help analysts in discovering requirements knowledge, i.e. high-level expressions containing functional requirements, preferences, constraints. Number 3 in Figure 1 highlights the need

[3] Use of images (e.g. expressing needs) and colours (e.g. red for complaining) for expressing some concepts.

of NL techniques that could be of usefulness and implemented in the framework, since these techniques might support the extraction of key terms from end-user feedback that will turn into hints to analysts and infer requirements knowledge. Then discovering candidate requirements to be changed or added. In this way, the candidate requirements will lead to the evolution of the existing specification of a software system. The acquired requirements knowledge can then be elaborated as goal-oriented models or BPMN specifications, to mention some options for representing it.

3 Research Methodology and Plan

Here below we present the methodology and high-level plan for two years PhD research. The methodology starts with the motivation of the research that is elaborated according to the reading of the state of the art and observation of the problem in managing end-user feedback. An overview of the main activities of the research plan is presented in Figure 2. In order to build a framework that contains the foundational concepts about end-user feedback, i.e. to answer *RQ1*, it is required to define the object of study, i.e. end-user feedback. To address *RQ2* the methodology also considers: i) the identification of current problems that one can encounter when collecting feedback (reported on the literature); and ii) the specification of candidate techniques that give solutions to such problems. The analysis of current end-user feedback and discussion to discover requirements knowledge concerns *RQ3*, along with the design of experiments for analysing feedback and therefore validation of them, i.e. *RQ4*. A relevant step in our research plan is the selection of a case of study for applying our preliminary findings, another step is to continue with a further investigation of the current feedback as expressed by end-users. We are currently investigating sampled data from Apache OpenOffice[4] bugzilla platform to explore and identify the existing dimensions in which feedback is expressed and compare those dimensions with the literature considered so far. Since we want to engage end-users and stimulate the acquisition of feedback, we have to investigate different types of feedback processes. We are considering, hence, to explore the use of open-source software (e.g. Liquidfeedback[5]) that could apply a specific type of feedback process (e.g. argumentation), then adapting such proposal to fit in our framework. Another activity is concentrated in studying and evaluating the

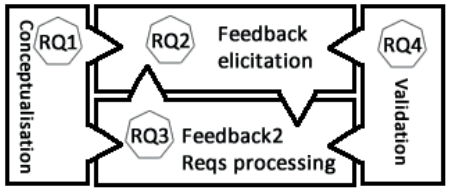

Fig. 2. Main activities in the plan that implements our research methodology.

[4] https://issues.apache.org/ooo/
[5] http://liquidfeedback.com/#Software

usefulness of the identification of written speech acts to discover important information in feedback. To evaluate our approach we are considering to perform survey research to know how is the perception of end-user feedback vs. our proposed conceptualisation. Besides this, the design and execution of a controlled experiment is being considered for assessing the engagement of users in providing feedback through an argumentation-based approach.

4 Progress

So far we have performed empirical studies over existing documentation of an already finished project, ACube, in the domain of health care [18, 17]. In this project the analysts carried out the requirements elicitation process by using HCI and RE techniques. The outcome of both investigations has given us clear indications that a multidisciplinary approach to elicit requirements is fruitful in terms of discovering and refining requirements knowledge, from several sources of information. One of such sources is a goal-oriented model which along with HCI techniques were applied in a collaborative and interdisciplinary requirements elicitation framework, thus presenting evidences of usefulness in identifying "criticalities" (i.e. relevant requirements). Nevertheless, the use of both HCI and RE techniques and methods were applied in face-to-face meetings with the involved stakeholders. On the other hand, a meta-model has been devised according to the characterisation in Section 2.1 (see [19] for a preliminary proposal), which is progressively consolidated. From our point of view, end-user feedback may have one or more purposes. *Purpose* is a conceptual entity that is specialised into clarification, correction, improvement, and strategic behaviour. *Improvement* is introduced to indicate what should be improved in the software application, i.e. a functionality, a quality improvement or the execution of a certain task.

Acknowledgements

I thank my advisor Anna Perini who has given me her invaluable feedback and support. As well as all the researchers from the SE unit in FBK and from the Doctoral Symposium for their feedback.

References

1. I. Arapakis, J. M. Jose, and P. D. Gray. Affective feedback: an investigation into the role of emotions in the information seeking process. In *SIGIR*, pages 395–402. ACM, 2008.
2. Y. Brun, G. Di Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw. Engineering self-adaptive systems through feedback loops. In *SE for Self-Adaptive Syst.*, pages 48–70. Springer-Verlag, Berlin, Heidelberg, 2009.
3. Björn Decker, Eric Ras, Jörg Rech, Pascal Jaubert, and Marco Rieth. Wiki-based stakeholder participation in requirements engineering. *IEEE Software*, 24(2):28–35, 2007.

4. M. Funk. *Model-driven Design of Self-observing Products*. PhD thesis, Eindhoven University of Technology, March 2011.
5. R. Gacitua, P. Sawyer, and V. Gervasi. Relevance-based abstraction identification: technique and evaluation. *Requir. Eng.*, 16(3):251–265, 2011.
6. M. W. Godfrey and D. M. German. The past, present, and future of software evolution. *Frontiers of Software Maintenance (FoSM'08)*, September 2008.
7. J. Hart, A. G. Sutcliffe, and A. De Angeli. Using affect to evaluate user engagement. In *CHI Extended Abstracts*, pages 1811–1834. ACM, 2012.
8. J. Hattie and H. Timperley. The power of feedback. *Review of Educational Research*, 77(1):81–112, March 2007.
9. IdeaScale. Ideascale. `http://ideascale.com/iphone/`, 2010.
10. UserVoice Inc. uservoice. `http://www.uservoice.com/`, 2012.
11. M. Johnson and S. Hyysalo. Lessons for participatory designers of social media: long-term user involvement strategies in industry. PDC '12, pages 71–80, New York, NY, USA, 2012. ACM.
12. I. Jureta, J. Mylopoulos, and S. Faulkner. Analysis of multi-party agreement in requirements validation. In *RE*, pages 57–66. IEEE Computer Society, 2009.
13. A. J. Ko and P. K. Chilana. How power users help and hinder open bug reporting. CHI '10, pages 1665–1674, New York, NY, USA, 2010. ACM.
14. W. Maalej, H-J Happel, and A. Rashid. When users become collaborators: towards continuous and context-aware user input. In *ACM SIGPLAN*, OOPSLA '09, pages 981–990, New York, NY, USA, 2009. ACM.
15. N. H. Madhavji, J. C. Fernández-Ramil, and D. E. Perry, editors. *Software Evolution and Feedback: Theory and Practice*. John Wiley and Sons Ltd, 2006.
16. S. Mohammad. Colourful language: measuring word-colour associations. CMCL '11, pages 97–106, Stroudsburg, PA, USA, 2011.
17. I. Morales-Ramirez, M. Vergne, M. Morandini, L. Sabatucci, A. Perini, and A. Susi. Revealing the obvious?: A retrospective artefact analysis for an ambient assisted-living project. In *2012 IEEE Second International Workshop (EmpiRE)*, pages 41–48, sep 2012.
18. I. Morales-Ramirez, M. Vergne, M. Morandini, L. Sabatucci, A. Perini, and A. Susi. Where did the requirements come from? a retrospective case study. In *ER Workshops*, volume 7518 of *LNCS*, pages 185–194. Springer, 2012.
19. M. Vergne, I. Morales-Ramirez, M. Morandini, A. Susi, and A. Perini. Analysing User Feedback and Finding Experts: Can Goal-Orientation Help? To appear in *iStar 2013 Workshop*.
20. Edna Holland Mory. Feedback research revisited. *Handbook of research on educational communications and technology*, 45(1):745–784, 2004.
21. N. A. Qureshi, N. Seyff, and A. Perini. Satisfying User Needs at the Right Time and in the Right Place: A Research Preview. In *REFSQ*, volume 6606 of *LNCS*, pages 94–99. Springer, 2011.
22. K. Schneider. Focusing spontaneous feedback to support system evolution. In *RE*, pages 165–174. IEEE, 2011.
23. N. Seyff, F. Graf, and N. Maiden. Using mobile re tools to give end-users their own voice. In *RE, 2010 18th IEEE International*, pages 37 –46, 27 2010-oct. 1 2010.
24. A. G. Sutcliffe, S. Thew, and P. Jarvis. Experience with user-centred requirements engineering. *Requir. Eng.*, 16(4):267–280, 2011.
25. Roel Wieringa. Design science as nested problem solving. In Vijay K. Vaishnavi and Sandeep Purao, editors, *DESRIST*. ACM, 2009.

A model-driven method to support conceptual model evolution

Marcela Ruiz

PROS Research Centre, Universitat Politècnica de València, Spain
lruiz@pros.upv.es

Abstract. Systems need to evolve to be adapted with changes in their contexts. There are several motivations to promote investment and scientific effort for specifying systems by means of conceptual models and supporting its evolu tion. As an example, the software engineering community is addressing solu tions for supporting model traceability, continuous improvement of business process, organisational reengineering, information system maintenance, etc. Model driven techniques have been developed in order to analyse systems rais ing the abstraction level of its specification. However, a support for conceptual model evolution by means of model driven techniques is still needed. In this thesis we propose a method that involves model driven capabilities for design ing and providing guidelines, techniques, and tools to support conceptual model evolution. We plan to apply our method to guide the evolution of a business process management suite. This way, we also provide mechanism to facilitate industrial adoption.

Keywords: conceptual model evolution, model driven capabilities, reengineer ing frameworks, model traceability, business process modelling, intentional modelling, requirements engineering, information system maintenance

1 Motivation

Since we are living in a changing world, systems are in continual revitalisation and evolution as a response of requirements and needs of their context. For instance, in organisational context, companies need to rethink business processes, infrastructures, technologies, resources, etc. according to new demands from their environment or changes in their organisational objectives. Business processes should also be transformed to support the new processes and tasks that result from the involvement of new objectives or goals in the organisation. Then, constant organisational change and its influence in processes and products must be considered as a fundamental rule of competitive strategy for continuous improvement [1]. Similarly, information technologies are part of the organisation's evolution. Both, software and hardware systems should be part of the improvement processes. Hardware systems are impacted by technological changes. For software systems, the high pressure of a very short time-to-market often forces developers to implement the code of the application directly, without using a disciplined development process, which may have disastrous effects

on the quality and documentation of the delivered software application [2]. These practices have been the motivation for opening new research lines in order to support post-delivery life-cycle activities. Besides, with regard to the keynote of the ERCIM News 88 magazine[1], some of external drivers for changing software are innovation, cost reduction and regulation; factors that need to be supported by techniques, tools and methods.

Furthermore, reengineering is a concept that has been widely used in software maintenance and information system evolution [4]. Reengineering is the process of reverse engineering to get higher level specifications, evolution of these specifications, and forward engineering to develop a new version of the software application [5]. Several MDD proposals and tools are currently aiming at supporting reengineering process and software maintenance for general scenarios. Also, the Object Management Group (OMG) is working on promote an industrial consensus on modernisation of existing applications by means of an initiative named Architecture-Driven Modernisation (ADM) [6]. This initiative is based on the MDD paradigm and it is implemented in an Eclipse-based tool named MoDisco [7]. However, full support by means of methods and tools for the evolution process is still an open challenge.

By reintroducing conceptual model evolution, the main goal of my PhD thesis is *to design a method that involves model driven capabilities in order to support conceptu al model evolution*. This way, we[2] design techniques, guidelines, and tools to evolve a current system to a desired system. The method provides conceptual model evolution with extensions for supporting business process evolution, goal-driven model evolution and a migration module that have into account reengineering scenarios. These extensions exemplify how to address conceptual model evolution in contexts as innovation, regulation, cost reduction (indicators), etc. As a result, the method is modular and can be used for different conceptual model evolution scenarios. Concretely we propose four modules: 1) an evolution module that integrates traceability support and reports of evolution process (this module is the foundation for the following modules); 2) a business module for supporting business process model evolution; 3) an intentional module for supporting goal-driven evolution (this module refers the introduction of indicators); and 4) a migration module for reengineering data systems.

The paper is structured as follows: Section 2 presents the problem statement that is confronted in this thesis. Section 3 describes a set of research questions. Section 4 presents our proposal: a model-driven method to support system evolution. Section 5 sketches the research methodology. Section 6 discusses the progress of the thesis, results, and future work.

---

[1] The ERCIM News 88 special theme was “Evolving Software” [3]. The magazine put to gether a set of papers to give an overview of both traditional and emerging software engi neering techniques, tools and approaches used by software evolution experts.

[2] This doctoral paper is about my PhD thesis. I am going to use the first plural person to acknowledge to my supervisors Sergio España and Óscar Pastor.

2 Problem statement

Traditionally in software system development, the evolution process and information system maintenance have been faced by means of the reengineering process [8]. For these reason, we explore current solutions in these fields in order to find related research that confronts conceptual model evolution.

The reengineering process is commonly defined and widely used by the scientific community by means of the metaphor of the "horseshoe" model, which purpose is to present the reengineering process in a figure (the horseshoe is basically a left-hand side, a right-hand side and a bridge between the sides). In general terms, the left-hand side of the horseshoe model consists of an extraction from an existing system to get the system specification, the right-hand side consist of conventional software development activities, and the bridge between the sides consists of a set of transformations from the old system to the new one [8]. Both, the left-hand side and right-hand side represent different levels of abstraction of the system. Nowadays, the Object Management Group (OMG) is working on promote an industrial consensus on modernisation of existing application by means of the initiative named Architecture-Driven Modernisation (ADM) [6]. This initiative is based on the MDD paradigm to automate the horseshoe model. Due to there are several modernisation projects, ADM establishes a general scenario for modernisation. This scenario analyses three major architectural perspectives (business architecture, application and data architecture, and technical architecture) and two domains (business and IT domain). However, full support for the evolution process (the bridge between the sides) is still missing.

The authors of [9] aimed to automate the horseshoe model, although it is not severely applied. The idea addresses the transformations between the both sides of the horseshoe model in a low level of abstraction. These transformations are parameterised with information from models of higher level of abstraction. The objective is to avoid losing information in the abstraction processes in both sides of the horseshoe model. Nevertheless, full support by means of methods and tools for the evolution process is still an open research challenge.

Due to we are looking for supporting the evolution of a business process management suit (BPMS) and this evolution process does not involve the complete reengineering scenario, we focus on the evolution process and we find that it is necessary to provide a support by means of a method that cover guidelines, techniques and tools to facilitate the evolution process. This is the open challenge that we face in this thesis. Although we have into account modules to support evolution in several cases as we explained in the Section 1.

3 Research questions

We follow design science to classify our research questions in knowledge problems (KP) and practical problems (PP) [10]. This way, we are looking for highlighting our research results by means of producing useful artefacts. This thesis is focused on con-

ceptual model evolution. To achieve the main goal, we conceive the following research questions:

- **RQ1** (KP). What elements are common in conceptual model evolution? The answer to this question should clarify terminology, stakeholders, and helps to establish a conceptual framework to facilitate reasoning about conceptual model evolution.
- **RQ2** (KP). Which are the current conceptual model evolution methods? The answer to this question should establish the state of the art about current conceptual model evolution support.
    - **RQ2.1** (KP). Which of these methods are model-driven oriented?
- **RQ3** (PP). How can be supported a conceptual model evolution method? The answer to this question refers to the main **goal** of this thesis.
    - **RQ3.1** (PP). What guidelines are needed in order to evolve conceptual models?
    - **RQ3.2** (PP). What techniques are needed in order to facilitate the use of the method?
    - **RQ3.3** (PP). What tools are needed in order to support the use of guidelines and techniques?
- **RQ4** (PP). How can possible scenarios be integrated in the conceptual model evolution method? The answer to this question refers the modules to support business process evolution, goal-driven evolution, and reengineering.
- **RQ5** (KP). How can the model-driven method to support conceptual model evolution be validated? The answer to this question should establish a validation framework to measure feasibility, trade-off and sensitivity.

4 Solutions and contributions

We face the design of the method by two main motivations: 1) Market pull or demand pull and 2) Technology push [11]. The first one refers our motivation to evolve the BPMS (a real case and we have into account the user needs). We call it market-driven solution. The second one refers our motivation to provide an invention without proper consideration of whether or not it satisfies a set of specific user needs. We call it technology push-driven solution.

To face conceptual model evolution, we have been inspired by the metaphor of a "horseshoe" of Kazman et. al. [8]. Carrying the horseshoe metaphor to the MDD field, an interesting evolution method can be provided for different scenarios. As a result, models are the main artefact and the analysis of them is in a high level of abstraction.

With regards to the main motivations for designing the method, we conceive the method with extensions (modules) for supporting the market-driven solution and the technology push-driven solution. As a result, the method is modular and can be used for different conceptual model evolution scenarios. Concretely we propose four modules: 1) a data system evolution module that integrates traceability support and reports

of evolution process (this module is the foundation for the following modules); 2) a business module for supporting business process model evolution; 3) an intentional module for supporting goal-driven evolution (this module refers the introduction of indicators); and 4) a migration module for reengineering systems. The modules 1) and 4) are addressing to fulfil the motivations of the market-driven solution. This solution refers the evolution of a BPMS. The modules 2) and 3) are a research effort in order to provide extensions that we consider interesting to be involved in possible real cases.

Fig. 1 presents an overview of the artefacts that are having into account in the conceptual model evolution method. Briefly, we present four modules together. The module of reengineering consists of three processes and four artefacts. The first process is the reverse engineering process; whose input is the first artefact, the As-Is system (previously referred as old system). The result of the reverse engineering process is the second artefact, the As-Is models (that represent the As-Is system in an abstract way). The second process is the evolution process; whose inputs are the As-Is models. As a result, the output of the evolution process is the third artefact, the To-Be models (evolved models). The third process is the forward engineering process; whose input is the To-Be models and the output is the fourth artefact, the To-Be system (system that results from the reengineering process and fulfils the new goals and needs of the organisation).

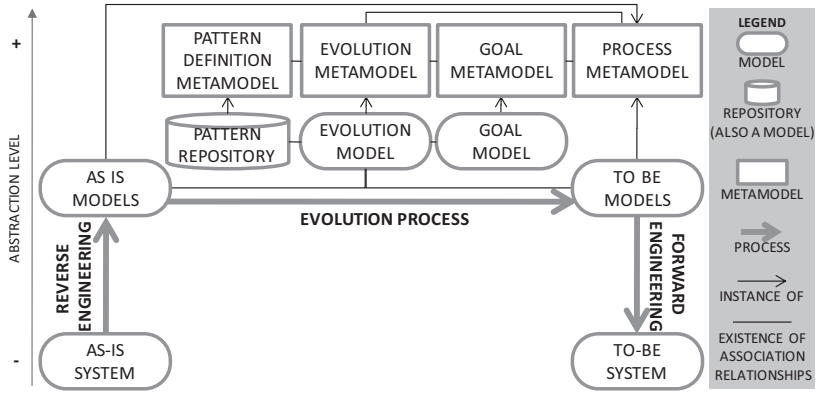

Fig. 1. Artefacts of the data system evolution method

With regards to the module of support for business process evolution, we found several works addressing he derivation of information systems from/to business processes (reverse engineering/forward engineering) [12-14]. We conceive business process models as important artefacts for representing organisational behaviour, our work focus on developing specific artefacts for evolving them. To support business process model evolution, we guide this evolution by means of pattern-driven and model-driven solution. This way, we propose a pattern definition metamodel to specify common behaviour. These patterns are stored in a repository for their use.

Concerning the conceptual model evolution support, we propose an evolution metamodel with several purposes: 1) to relate patterns with the models that are being evolved; 2) to relate the evolution process with the organisational goal that motivate the evolution process; and finally, 3) to establish traceability information in order to generate logs or reports about model evolution.

Concerning the goal-driven module, a goal metamodel and the process metamodel are artefacts to analyse information systems from different levels of abstraction, we propose to carry out an alignment between these perspectives to relate the process perspective with the intentional/goal perspective.

5 Research methodology

This PhD project follows the design science framework to design a new artefact: a model-driven method to support conceptual model evolution. The research methodology is explained by means of regulative cycles that were conceived in order to answer the research questions. Fig. 2 presents the research methodology.

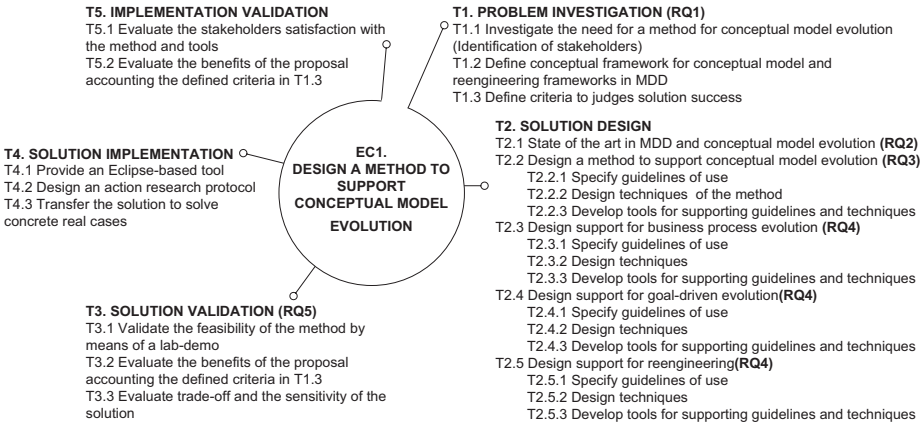

Fig. 2. Overview of the research methodology

Since our proposal focus on the development of a new artefact, the main cycle of the research methodology is an engineering cycle (EC1. Design a model driven method to support data system evolution). Concretely, this cycle is formed by 5 main tasks: T1) problem investigation; T2) solution design; T3) solution validation; T4) solution implementation; and T5) implementation validation.

A SME needs evolve it business process management suit (BPMS). Since the BPMS is specified by means of models, we investigate current research to support conceptual model evolution. We identify the stakeholders or possible users of the method. To define the problem and define the method, we provide a conceptual framework to avoid terminology incoherence. In addition, we establish the criteria to

judge the solution success when we finish the engineering cycle. These activities are related to T1.

In T2 we explore available solutions by reviewing state of the art. We design a new solution; i.e. our method. To do that, we design the guidelines of use; we provide techniques to facilitate the use of the method; and we develop tools (prototypes built in the laboratory) to support guidelines and techniques. Also, we design the support for the modules of business process evolution, goal-driven evolution and reengineering frameworks.

The method is validated in T3. We demonstrate the feasibility by means of lab-demo. We establish a comparative with the results of the lab-demo with the criteria defined in T1.3. Also, we evaluate trade-off and sensitivity of the solution.

In T4 we implement the method using Eclipse based tools, design an action research protocol to transfer the solution to the SME in order to evolve the BPMS. Finally, in T5 we assess the operability of the tool, stakeholder's satisfaction and criteria of success by means the results of the action research protocol carried out in T4.

6 Progress of the thesis

In 2012, we studied organisational reengineering framework, focusing on RQ1 and RQ2. Furthermore, we explored the alignment between the process and the goal perspectives. As a proof of concept, we have aligned the i* framework with the Communication Analysis modelling techniques. This proof of concept refers the RQ4. Also, we implemented the alignment of this modelling languages in an Eclipse-based tool (this implementation refers RQ3.). And we analysed the benefits and the limitations of aligning process and goal perspectives. We started a first version of the definition of the artefacts to support model evolution (Traceability support).

In 2013, we plan to build the conceptual framework for the method. We are going to establish a pattern model repository (RQ4). In addition, we are looking for implementing pattern definition metamodel and evolution metamodel in an Eclipse plug-in (RQ3). We plan to implement a support for model evolution and specification of reports and documentation about system evolution. In addition, we are going to explore the support for system interoperability and reengineering (RQ4).

In 2014-2015 we plan to establish the guidelines of use. We plan to validate the method and the prototype by means of laboratory demos. Afterward, we plan to carry out action research to guide the evolution of a business process management suit (BPMS). The idea is to estimate scalability, trade-off and sensitivity of our method. This validation refers RQ5.

We plan to finalize the implementation and the implementation validation of the method in 2015.

Acknowledgments

I acknowledge to my supervisors Sergio España and Óscar Pastor for their invaluable support and advices to drive my thesis.

We acknowledge the support of the Spanish Ministry of Science and Innovation project PROS-Req (TIN2010-19130- C02-02), the Generalitat Valenciana project ORCA (PROMETEO/2009/015), the Santiago Grisolía grant (GRISOLIA/2011/005) and the ERDF structural funds.

References

[1] A. P. Sage, "Systems Engineering and Systems Management for Reengineering," *Systems Software,* pp. 3 25, 1995.

[2] G. A. Di Lucca*, et al.*, "Reverse engineering Web applications: the WARE approach," *Journal of software maintenance and evolution: Research and practice,* vol. 16, pp. 71 101, 2004.

[3] J. Visser. (2012) Change is the constant. *ERCIM news Special theme: Evolving Software*. 3.

[4] I. García Rodíguez de Guzmán*, et al.*, "An integrated environment for reengineering," presented at the 21 st IEEE International Conference on Software Maintenance (ICSM'05), 2005.

[5] R. S. Arnold, *Software Reengineering*: IEEE Press, 1992.

[6] OMG. (2012, *Architecture Driven Modernization (ADM)*. Available: http://adm.omg.org/

[7] OMG. (2012, *MoDisco*. Available: http://www.eclipse.org/MoDisco/

[8] R. Kazman*, et al.*, "Requirements for Integrating Software Architecture and Reengineering Models: CORUM II," presented at the Working Conference on Reverse Engineering (WCRE 1998), 1998.

[9] J. Sánchez Cuadrado*, et al.*, "Parametrización de las transformaciones horizontales en el modelo de herradura," presented at the Jornadas de Ingeniería de Software y Bases de Datos (JISBD'12), Almería, Spain, 2012.

[10] R. Wieringa, "Design Science as Nested Problem Solving," presented at the 4th International Conference In Design Science Research In Information System and Technology (DESRIST'09), Malvern, PA, USA, 2009.

[11] M. J. C. Martin, *Managing Innovation and Entrepreneurship in Technology Based Firms* vol. 43, 1996.

[12] J. L. de la Vara, "Business Process Based Requirements Specification and Object Oriented Conceptual Modelling of Information Systems," Departamento de Sistemas Informáticos y Computación (DSIC), Universitat Polièc nica de València, Valencia, Spain, 2011.

[13] S. España, "Methodological integration of Communication Analysis into a Model Driven software development framework," PhD. Computer Science, Departamento de Sistemas Informáticos y Computación (DSIC), Universitat Politècnica de València, Valencia, 2011.

[14] O. Pastor and J. C. Molina, *Model Driven Architecture in practice: a software production environment based on conceptual modeling*. New York: Springer, 2007.

Part IV

REFSQ 2013 Poster Session Proceedings

11 Preface

Editors

Joerg Doerr
Fraunhofer IESE, Germany, joerg.doerr@iese.fraunhofer.de

Andreas L Opdahl
Univ. of Bergen, Norway, andreas.opdahl@uib.no

Foreword to the Research Poster Abstracts

REFSQ 2013 for the first time included displays and brief oral presentations of regular research posters in addition to posters from the Doctoral Symposium and the Empirical Fair. One group of research posters were invited from regular submissions that were considered promising by the Program Committee, but not yet quite ready for the paper sessions. Another group of posters were received as extended abstracts after a separate Call for Posters, distributed on the usual emailing lists. The latter group were carefully checked for originality and research content by selected REFSQ 2013 PC Members.

In the end, nine regular research posters were on display in the coffee hall during breaks and inside the conference rooms, along with posters of the Doctoral Symposium and the Empirical Fair. All poster authors were also invited to give one-minute presentations of their posters during the first break on the first conference day. After the conference, the authors were invited to revise their poster abstracts for inclusion in these Post-Conference Proceedings. You will find the extended abstracts for the nine regular research posters on the following pages.

We would like to thank the organisers for working hard to deal with all the practical poster arrangements!

Joerg Doerr, Andreas L Opdahl

Program Chairs, REFSQ 2013

12 Poster Session

Poster Session Programme

Formalizing Requirements Engineering by a "Requirements Engineering House" 247
Sebastian Adam, Norman Riegel, Anne Hess, Özgür Ünalan, Simon Darting, Andreas Maier and Karina Villela

Managing variability in an automotive company: Bridging the gap between PLE and MBSE 249
Cosmin Dumitrescu, Camille Salinesi and Alain Dauron

Investigating Requirements Communication in Iterative Projects 251
Anne Hess, Norbert Seyff, Emmerich Fuchs and Daniel Rapp

Representation of Requirements Changes to Stakeholders in Self-adaptive Systems 253
Fabian Kneer and Erik Kamsties

Requirements Engineering and Building Construction: Requirements Engineering for a Synagogue Kitchen with Use Cases and Scenarios 255
Cyril Mauger and Daniel Berry

An Experimentation Platform to Literally Support RE Tasks 257
Thorsten Merten, Kim Lauenroth and Simone Bürsner

Group Versus Individual Use of an Optimized and the Full EPMcreate as Creativity Enhancement Techniques for Web Site Requirements Elicitation 259
Victoria Sakhnini, Luisa Mich and Daniel Berry

Security requirements analysis based on security and domain ontologies 261
Amina Souag, Camille Salinesi and Isabelle Wattiau

The Recommendation in the Product Line Configuration Process 265
Raouia Triki and Camille Salinesi

Formalizing Requirements Engineering by a "Requirements Engineering House"

Sebastian Adam, Norman Riegel, Anne Hess, Özgür Ünalan, Simon Darting, Andreas Maier, Karina Villela

[1]Fraunhofer IESE, Fraunhofer Platz 1, 67663 Kaiserslautern
{sebastian.adam, norman.riegel, anne.hess, oezguer.uenalan, simon.darting, andreas.maier, karina.villela}@iese.fraunhofer.de

It is widely acknowledged that there exists no "one-fits-all" Requirements Engineering (RE) process. Thus, at the beginning of a project, decisions have to be made regarding the following four dimensions: *artifacts* to produce, *practices* to follow, *stakeholder* groups to involve, and *requirement types* to be covered.

We have observed in several industry projects that practitioners often face challenges when making such decisions. The reasons are:

- Practitioners typically lack an overview of these four dimensions, and they often focus on concrete practices rather than taking into consideration the relevance of artifacts, stakeholders, and different requirement types.
- Knowledge about the interdependencies between the four dimensions, which is crucial for a proper decision making, is still implicit knowledge of RE experts and not sufficiently described in literature yet. However, this knowledge is indispensable to elaborate a precisely tailored RE process for the given project context.

As a first step towards addressing these challenges, we have defined a set of conceptual models, which are organized in a so-called "Requirements Engineering House". These models describe the concepts to be mastered when defining a RE process for a specific project and also show how those concepts are related. The idea of the RE house was taken from the ARIS house [1], in which different aspects of a business process are clearly separated into four dimensions: organizational units which are responsible for conducting a process, data to be processed in a process, activities to be performed in a process, and results / products to be produced by a process.

Similarly, the RE house acts as a framework for concrete models in several dimensions (see Figure 1).

Stakeholders and Roles: This model is intended to clarify which stakeholders should be involved in requirements elicitation activities in order to assure that all relevant requirements will be included. Furthermore, this model also captures important engineering roles in downstream activities and their respective tasks. Those tasks will later "consume" the specified requirements and thus have great influence on the requirements artifacts to be produced.

Reference Objects: This model is intended to explain for which conceptual elements (e.g., business processes, human-system-interactions, system functions, data) concrete requirements have to be elicited in a project. Furthermore, this model defines assessment criteria for these conceptual elements in order to make requirements prioritization more objective. For example, business processes can be prioritized based on their value for the overall organization.

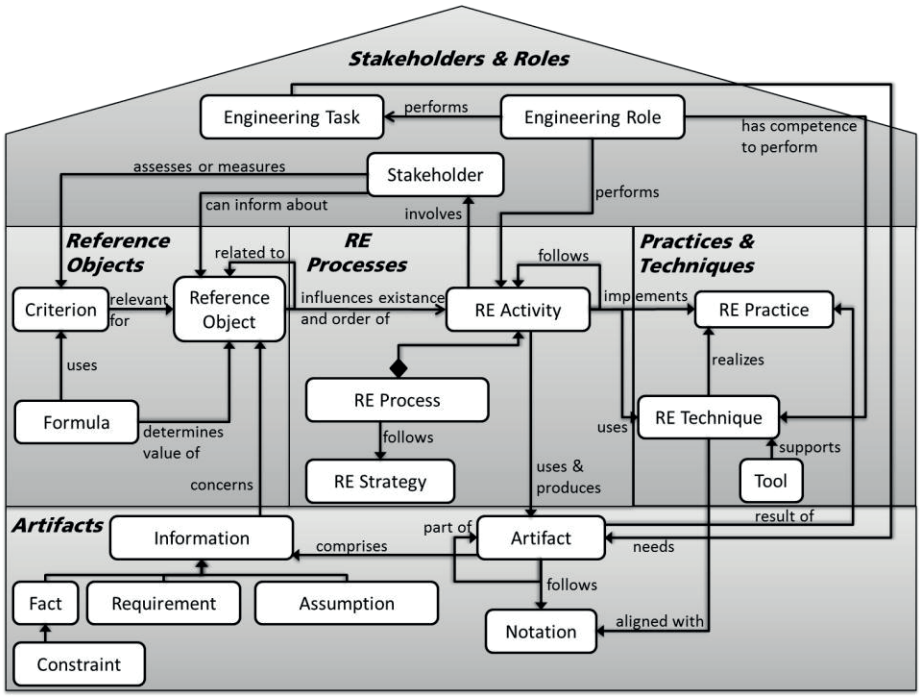

Figure 1. Meta Model of the RE House

Practices and Techniques: This model is intended to describe requirements engineering best practices, e.g., according to the good practice guide [2] or ReqMan [3], and concrete techniques to implement these practices.

Artifacts: This model is intended to capture how elicited requirements concerning a certain reference object should be specified (e.g., a human-system-interaction should be described in the form of a use case description). Furthermore, this model provides concrete notations that can be used to specify each artifact.

RE Processes: This model is intended to define different "templates" for requirements processes implementing certain strategies such as waterfall-like RE, iterative RE, or agile RE.

Finally, our purpose is to use the aforementioned models to support (semi-) automated derivation of RE processes. Therefore, after completely defining the RE house's models, we will start defining transformation rules and tailoring tools.

References

[1] Scheer, A.-W.: ARIS - Modellierungsmethoden, Metamodelle, Anwendungen. Springer, Berlin (1998)

[2] Sommerville, I., Sawyer, P.: Requirements Engineering – A good practice guide. John Wiley, New York (1997)

[3] Doerr, J., Koenig, T., Olsson, T., & Adam, S.: Das ReqMan Prozessrahmenwerk. IESE-Report Nr, 141. (2006)

Managing variability in an automotive company: Bridging the gap between PLE and MBSE

Cosmin Dumitrescu[1,2], Camille Salinesi[1], and Alain Dauron[2]

[1] Centre de Recherche en Informatique,
Université Paris 1 Panthéon-Sorbonne
90 rue de Tolbiac, 75013 Paris, France
camille.salinesi@univ-paris1.fr
cosmin.dumitrescu@malix.univ-paris1.fr
[2] Technocentre Renault,
1 avenue du Golf, 78288 Guyancourt, France
alain.dauron@renault.com

Keywords: Model Based Systems Engineering, Product Line Engineering, variability modeling

One of the salient problems of model based systems engineering is to bridge the gap between customer features and technical features. "Features" being the central concept of PL engineering notations, we need to specify both kinds of features, as well as the dependencies between them. The main difficulty lies in the conceptual mismatch between the features coming from different types of sources.

While on the other hand, systems features in the user or customer view (we call it marketing view) lie in terms of their goals, tasks, use cases, or high level user visible functions and components. The engineering view is more concerned about technical components, most often invisible to the customer, their structure, and how they deliver functions that comply with interfaces or internal requirements.

The second difficulty is related to the configuration process: how to specify all systems features in a way that lets users and engineers configure independently. Two kinds of variability emerge: systems variability, and variability in the way to make configuration decisions. These need to be specified in separation to avoid mix of concern issues.

As is the case for many companies that produce large complex systems, our purpose is to integrate variability management with model based systems engineering (MBSE). Consistently with other domains, some features need to be visible on the organization level, and thus the expression of these features requires compatibility with other views (such as supply chain and manufacturing) and needs to make abstraction of specific aspects of MBSE models (for example SysML). We believe that in order to handle this issue, two questions need to be addressed : (i) how to represent complex constraints, and (ii) how to separate among various kinds of variability.

Our approach follows the work of Pohl et al. [5], proposing an extension to modeling variability orthogonally (OVM), by adopting concepts that further detail the way variability is documented for systems and components, and at the

same time supports our methodological framework , by introducing: sources for variability, views, types of variability, impact on the system architecture, selection criteria based on product properties. Furthermore, we focus on constraints, where some variability modeling techniques have limited expressivity and often require introducing auxiliary elements, either to create complex constraint expressions, or to structure the model . We aim to represent constraints in the user (marketing) view, the technical (architecture) views, and in the relation (binding) between features and system elements, but also provide enough flexibility to surface a minimum amount of information from low level, technical constraints to the customer visible variability constraints. Meanwhile, we rely on the modeling technique defined by Tessier et al. [6] as an intermediate layer of abstraction for constraints embedded in UML/SysML models, and on the work of Mazo et al. [4] for variability constraints validation and improvements in configuration interactivity through constraint-based recommendation.
In general, variability modeling in UML/SysML models relies on one of these three techniques: (i) exploitation of the semantics of the base model, (ii) using an external model for the management of variability of the base model (for example using a third party software to manage SysML models) and (ii) adding new semantics and concepts to the base model, which is also the approach we have taken.
Integration of variability management in MBSE, was a first step to bridge the gap between customer oriented features and the diversity of components, thorough systems engineering. We intend to focus future work on improving the configuration process through constraint based recommendation (faster response times, ensure validity, reduce number of configuration steps etc.) and multi-objective system optimization [2].

References

1. Astesana, J.-M.; Cosserat, L.; Fargier, H., "Constraint-based Vehicle Configuration: A Case Study," Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on , vol.1, no., pp.68,75, 27–29 Oct. 2010
2. Doufene, A., Chalé Góngora, H., Krob, D.: Complex Systems Architecture Framework. Extension to Multi-Objective Optimization. Presented at the Complex Systems Design & Management 2012 , Paris December (2012).
3. Dumitrescu, C., Tessier, P., Salinesi, C., Grard, S., Dauron, A.: Flexible Product Line Derivation applied to a Model Based Systems Engineering process. Presented at the Complex Systems Design & Management 2012 , Paris December (2012).
4. Mazo, R., Salinesi, C., Diaz, D., Djebbi, O., Lora-Michiels, A.: Constraints: The heart of domain and application engineering in the product lines engineering strategy. International Journal of Information System Modeling and Design (IJISMD). 3, 33-68 (2012).
5. Pohl, K., Böckle, G., Linden, F.J. van der: Software Product Line Engineering: Foundations, Principles and Techniques. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2005).
6. Tessier, P., Gérard, S., Terrier, F., Geib, J.M.: Using variation propagation for model-driven management of a system family. Software Product Lines. 222-233 (2005).

Investigating Requirements Communication in Iterative Projects

Anne Hess[1], Norbert Seyff[2], Emmerich Fuchs[3], Daniel Rapp[3]

[1] Fraunhofer IESE, 67663 Kaiserslautern, Germany,
Anne.Hess@iese.fraunhofer.de
[2] University of Zurich, Department of Informatics, 8050 Zurich, Switzerland
seyff@ifi.uzh.ch
[3] Zühlke Management Consultants, 8952 Schlieren (Zurich), Switzerland
{efu,dra}@zuehlke.com

1 Research Content

It is widely known that requirements specifications play a crucial role in software engineering as these specifications serve as a source of information for a variety of roles involved in a development project. Therefore specifications have to address and satisfy different information and communication needs [3] [4].

We claim that addressing such needs by providing the right information, at the right level of detail, at the right point in time is a very challenging task – especially in iterative software development projects where development activities are highly interlaced.

Since its introduction in 1986, iterative software engineering (SE) has been established and introduced in many companies. There are indeed several practitioners and SE companies who successfully follow iterative SE approaches. However, our experiences and observations indicate that there are still a significant number of practitioners for whom performing RE in iterative projects successfully is still a challenge as often requirements artifacts are documented and communicated in a rather "monolithic" and waterfall-like approach.

Today, several prominent process models and guidelines have been established that support companies in introducing and following iterative SE, such as the "Spiral Model" [1] or the "Rational Unified Process" (RUP) [2] [6], etc. These guidelines also highlight the continuous communication of defined artifacts between different development disciplines such as software architecture, testing, etc. Moreover, for each discipline, they also explicitly specify the required input artifacts as well as the output artifacts, such as business use case models, business object models, and stakeholder requests.

However, an initial literature review indicates that current guidelines do not explicitly consider different project settings and resulting information needs. It is unclear at what level of detail an artifact needs to be communicated at a certain time and for whom it is relevant. The criticality of this communication becomes even more obvious in approaches discussing the important interrelation between requirements artifacts and early decisions towards possible solution variants made on software and system architecture levels, as discussed, for instance, in the "Twin Peaks Model" [5].

Initial studies indicate that the information needs within a particular project are strongly dependent on various factors such as the role, particular task and expertise of systems and / or development engineers, the project scope, and the domain [3] [4]. This variety might be one reason why it is challenging for practitioners to perform RE in iterative projects. Also, people's personal preferences might have an influence on what kind of information should be available when.

As part of our ongoing research we aim at investigating in more detail which RE artifacts need to be communicated when in order to satisfy the information needs of "requirements consumers" in iterative projects.

For this purpose, we are currently conducting a case study with more than 130 students who are enrolled in a practical SE course. Within this course, the students are divided into development teams of about 7 students each with predefined roles (i.e., requirements engineers, UI designers, testers, etc.). Each team is running a SE project motivated by a problem given by a real customer from industry. Altogether, there are 19 SE projects running in parallel, with each project following an iterative SE process. By means of interviews, questionnaires, discussions, observations, and retrospective analysis of activities and RE artifacts within the particular student project teams, we aim to elaborate an "artifact landscape" reflecting the communication flows between the different roles (i.e., which requirements artifacts are required for which role at what level of detail for which task).

Moreover, in the future we also aim to get insights into existing problems, reasons and their consequences related to RE in iterative projects by means of studies conducted in industry that will finally be used to suggest evaluation and assessment strategies for iterative software projects.

References

1. Boehm, B.: A Spiral Model of Software Development and Enhancement. SIGSOFT Software Engineering Notes 11, 4, 14 24. (1985)
2. Rational Software Whitepaper: Rational Unified Process, Best Practices for Software Development Teams, http://www.ibm.com/developerworks/rational/library/253.html
3. Gross, A[1]., Doerr, J.: What you need is what you get! The Vision of View based Requirements Specifications, Proceedings of 20th IEEE International Requirements Engineering Conference (RE'12) (2012)
4. Gross, A[1]., Doerr, J.: What do Software Architects Expect from Requirements Specifications? Int. Worksh. on the Twin Peaks of Requirements and Architecture (2012)
5. Nuseibeh, B.: Weaving The Software Development Process Between Requirements and Architecture". In Proceedings of ICSE2001 STRAW 01 (2001)
6. Kroll, P., Kruchten, P.: The Rational Unified Process Made Easy A Practitioners Guide to the RUP, Addison Wesley (2003)

[1] Same author as Hess, A.

Representation of Requirements Changes to Stakeholders in Self-adaptive Systems

Fabian Kneer
Erik Kamsties
Fachhochschule Dortmund
Emil-Figge-Straße 42
D-44227 Dortmund, Germany
{fabian.kneer@fh-dortmund.de, erik.kamsties@fh-dortmund.de}

1. Introduction

"Requirements at Runtime" is a novel area of requirements engineering [1]. It has a large research potential but many problems are not solved yet. Usually, a dynamically self-adaptive system does not have all information about its context. Because of this lack it has to change its behavior during runtime. There are many research approaches aiming at a better understanding of the own requirements of the system and thus provide a better handling of new events.

Usually, a user does not get any information about the adaption, but is just faced with the changed behavior. Additionally, a requirements engineer does not get feedback about adaptations in the field. There is an information gap between the orthodox requirements engineering and the dynamic requirements model.

We want to bridge this gap with a new runtime component, which traces the adaptations to their original requirements. With this component the user and the requirements engineer can get a better feedback about their requirements and how they change during the runtime.

In this poster we explain the problem in detail and the benefits from our solution. If the requirement engineer receives a feedback about requirements and how they change during the runtime, she can find for example incorrectly elicited requirements and better understand the true requirements. The user develops more trust in the system.

2. Poster

Figure 1 shows the principle steps which happen during the development of system (following [1] we call it *orthodox* requirements engineering) and what happen during the runtime. At first the requirements are developed in orthodox requirements engineering (left side of the figure). These requirements must be transformed into a runtime representation of the requirements model. This model is often represented in i*, KAOS, or a modified version of these techniques (e.g., [2],[3]). With the help of this model a monitor (e.g., based on Fuzzy logic) is configured. During runtime, the monitor collects information about the environment or feedback form the users.

Changes in the context of the system can thus be detected. Now the system must consider the potential impacts that these changes will bring, that is which requirements are involved. If a change of the model takes place the system must be changed as well. This can be an adaption or a new configuration to handle the change in the context.

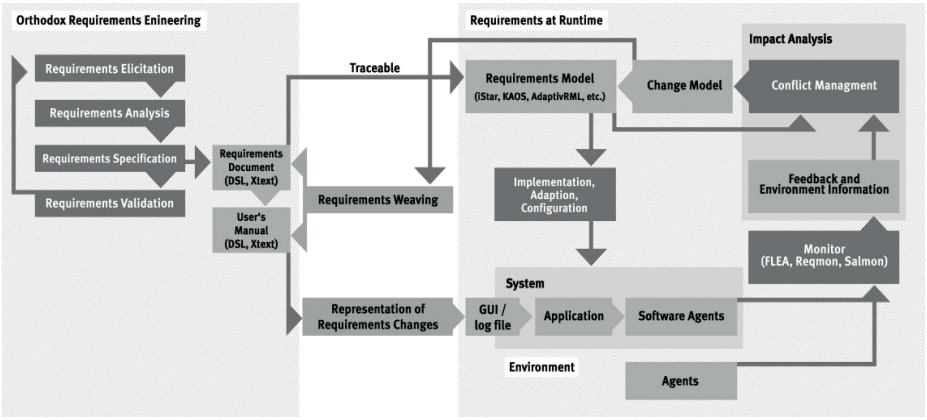

Fig. 1. Establishing the Link between Orthodox RE and Requirements at Runtime

At this point a new component is needed to collect the changes of the model and connect these to the original requirements found in the orthodox requirements engineering. This information can be used for feedbacks to the requirements engineers and the users of the system. With this feedback the users get a better understanding of the system and the requirements engineers get a better understanding of the context in which the system run. Our work is in line with Berry et al. [4] who defines four Levels of Requirements Engineering for and in Dynamic Adaptive System.

References

1. P. Sawyer, N. Bencomo, J. Whittle, E. Letier, A. Finkelstein, Requirements Aware Systems: A Research Agenda for RE for Self adaptive Systems, 18th IEEE International Requirements Engineering Conference (RE), 2010.
2. N. A. Qureshi, I. J. Jureta, A. Perini, Towards a Requirements Modeling Language for Self Adaptive Systems, Requirements Engineering: Foundation for Software Quality, 2012, pp 263 279
3. K. Welsh, P. Sawyer, N. Bencomo, Towards requirements aware systems: Run time resolution of design time assumptions, 26th IEEE/ACM International Conference, 2011
4. D. M. Berry, B. H. C. Cheng, J. Zhang, The Four Levels of Requirements Engineering for and in Dynamic Adaptive Systems, 11th International Workshop on Requirements Engineering Foundation for Software Quality (REFSQ), 2005

Requirements Engineering and Building Construction: Requirements Engineering for a Synagogue Kitchen with Use Cases and Scenarios

Cyril Mauger[1,2] and Daniel Berry[3]

[1] Service Science & Innovation Department, Public Research Centre Henri Tudor
29 Ave. John F. Kennedy, Luxembourg, Luxembourg
cyril.mauger@tudor.lu
[2] Laboratoire de Conception Fabrication Commande, Arts et Métiers ParisTech
4 Rue Augustin Fresnel, Metz, France
[3] Cheriton School of Computer Science
University of Waterloo
200 University Ave. West, Waterloo, Ontario N2L 3G1, Canada
dberry@uwaterloo.ca

Abstract

[Context and Motivation] Use cases and scenarios (UCaSs) are used in Requirements Engineering (RE) to illustrate a system's interactions with its users' roles to achieve the users' functional goals. UCaSs help achieve completeness in the specification of the system's requirements, to achieve an alignment between the needs of the system's client and the ultimate implemented system.

[Question/Problem] Are UCaSs and other RE techniques applicable to the requirements analysis for building construction?

[Principal Ideas/Results] This paper describes an experience in applying use UCaSs to help determine the requirements for a synagogue kitchen. The authors conducted a use-case-and-scenario-driven requirements analysis based on the original kitchen plan produced by a professional architect. From the difficulties in the plan exposed by the UCaSs, it became apparent that, as in the second author's earlier experiences remodeling one house and building another house, the professional architect's elicitation of functional requirements for the kitchen was inadequate.

Application of UCaSs to and a flow analysis of the original plan allowed the authors to produce an improved plan for the kitchen and to demonstrate to the synagogue kitchen's client why the improved plan is better for the kitchen's functional requirements.

Nevertheless, for reasons, probably emotional, that are not entirely clear, the client declined to use the new plan created by the not-professional-architects authors. He said that he would be sticking with the architect's original plan.

[Contribution] Lessons learned from the case study are:

- Techniques that work for RE for computer-based systems seem to work also for in the Architecture, Engineering, and Construction (AEC) domain.
- The AEC trade can learn about requirements analysis, what it calls "programming", from the RE field.
- The Software Engineering trade can learn about charging its clients additional fees in response client-initiated requirements creep.
- Even in a case in which functional requirements should clearly dominate a client's decisions, emotional issues may be entering into these decisions.
- The education of the typical AEC architect seems to be lacking a course in programming (i.e., RE), programming techniques such as UCaSs, and interacting with clients.

Keywords: Building construction, Building layout, Floor plans, Requirements specification, Scenarios, Use cases

The full paper is available at `se.uwaterloo.ca/~dberry/FTP_SITE/tech.reports/MaugerBerryKitchen2013.pdf`

An Experimentation Platform to Literally Support RE Tasks

Thorsten Merten[1], Kim Lauenroth[2], and Simone Bürsner[1]

[1] Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin
[2] adesso AG, Dortmund

Requirements Engineering (RE) requires a deep understanding on how different requirement artifacts can be used together. At REFSQ 2010 Ian Alexander emphasized that in this "Requirements Jigsaw-Puzzle" "requirements [e.g. different requirement artifact types, templates, etc.] are still not being put together well. Something is going wrong" [1]. RE practitioners are confronted with questions like: *How could I go on with my specification? What aspects of the problem do I still need to understand? How can I describe this and how what else do I need to relate to this?*

We know that short feedback loops are an important aspect in learning. They confront the learner with the consequences of his/her own activities. However, feedback loops are a challenge in RE since the period between specification and implementation is long. Short feedback loops based on the implementation of the requirements are therefore difficult to achieve. In order to improve the feedback situation for RE other means are necessary. A common industrial praxis is to provide feedback by means of reviews and coaching activities which shorten the feedback period. The feedback cycle could be shortened even more by assisting during the specification activities. Our current and future work to provide immediate feedback is to integrate different types of corresponding mechanisms in requirement management tools.

The proposed poster shows a) the current status of the *Redmine RE Plugin*[3], b) the support by our *RE Assistant* [2], c) preliminary evaluations of the *RE Assistant* and d) future work we would like to discuss. The preliminary evaluation with two groups of eight and ten master students has shown that the assistant can help understanding how to play the *requirements jigsaw* as well as speed up the process of creating the software requirement specification (SRS). Rule-based techniques allow us to specify differently formed pieces of the puzzle whereas the assistant helps the user to solve this puzzle (see Figure 1). Emphasizing on the aspect that our approach is not intended to provide any formal verification, the hints and best practices are based on traces between different RE artifacts and the data entered in the RE artifact templates. It therefore extends existing heuristic approaches like HeRA [3]. However, so far our approach does not give any orientation on abstraction levels and the question *What else do I need to understand?*.

Existing requirement management tools (the *Redmine RE Plugin* included) focus on displaying a *document oriented view*, simply because this is the way

[3] http://korem.de/redmine-re-plugin.html

specifications have always been written and perceived. The focus is therefore on completing the document and/or templates and not necessarily on problem comprehension. However, using a computer as the medium of interaction with the SRS, things can change.

Our idea is to integrate the rule based assistant with different *abstraction and understanding orientated perspectives* on the SRS. Although the idea of using different perspectives (or viewpoints) on the RE problem is not new (e.g. [4]) it has not been used to support users understanding the way we are experimenting. We think that an abstraction and problem/requirement/solution based workflow can help to understand requirements and their implied solution decisions on different abstraction levels better (cf. Figure 2).

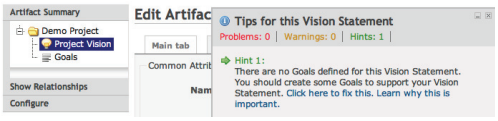

Fig. 1. RE Assistant Checking a SRS for Traces

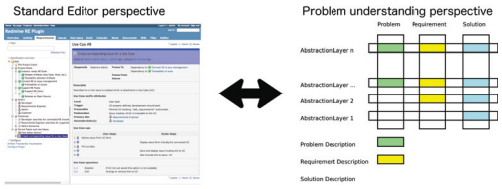

Fig. 2. Different Perspectives on a SRS

References

1. Alexander, I.: Keynote talk piecing together the requirements jigsaw-puzzle. In Wieringa, R., Persson, A., eds.: Requirements Engineering: Foundation for Software Quality. Volume 6182 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2010) 1–1
2. Merten, T., Schäfer, T., Bürsner, S.: Using RE knowledge to assist automatically during requirement specification. In: Seventh International Workshop on Requirements Engineering Education and Training (REET). (2012) 9–13
3. Knauss, E., Lübke, D., Meyer, S.: Feedback-driven requirements engineering: The heuristic requirements assistant. In: Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on. (2009) 587 – 590
4. Kotonya, G., Sommerville, I.: Requirements engineering with viewpoints. Software Engineering Journal **11**(1) (1 1996) 5–18

Group Versus Individual Use of an Optimized and the Full EPMcreate as Creativity Enhancement Techniques for Web Site Requirements Elicitation

Victoria Sakhnini[1], Luisa Mich[2], and Daniel M. Berry[1]

[1] Cheriton School of Computer Science, University of Waterloo
Waterloo, ON, N2L 3G1 Canada
vsakhnini@gmail.com, dberry@uwaterloo.ca

[2] Department of Industrial Engineering, University of Trento
I-38122 Trento, Italy
luisa.mich@unitn.it

Abstract

[Context] Creativity is often needed in requirements elicitation, e.g., in generating ideas for requirements. Techniques to enhance creativity are believed to be useful. In our research, we have been investigating a family of techniques based on the EPMcreate (Elementary Pragmatic Model Creative Requirements Engineering [A] TEchnique), which systematically exercises all sixteen combinations of the viewpoints of a pair of stakeholders (Fig. 1 & 2) [1, 2].

[Question] "How does the effectiveness of individuals in RE activities compare with that of groups?" is a general question in RE. We focused on how the size of a group affects the effectiveness of the group and the effectiveness of the group's members in generating requirement ideas using the full 16-step EPMcreate and using the optimized, 4-step POEPMcreate (Fig. 3).

[Method] We carried out a number of experiments in which individuals and groups of size two and four used EPMcreare and POEPMcreate to generate ideas for requirements for enhancing a Web site. Later, for triangulation, we conducted a survey to determine software development practitioners' attitudes on the comparison of the effectiveness of individuals and groups in requirements elicitation for real projects.

[Results] The data of the experiments indicate that the size of a group using EPMcreate and POEPMcreate does affect the number of raw and new requirement ideas generated by the group and by each member of the group. The larger a group is, the more raw and new requirement ideas it generates (Fig. 4). However, the smaller a group is, the more raw and new requirement ideas the average of its members generates (Fig. 5). The survey results indicate that experienced software development practitioners have observed the same and seem to act accordingly (Fig. 6) [3].

References

1. Mich, L., Anesi, C., Berry, D.M.: Applying a pragmatics-based creativity-fostering technique to requirements elicitation. Requirements Engineering Journal **10** (2005) 262–274
2. Sakhnini, V., Mich, L., Berry, D.M.: The effectiveness of an optimized EPMcreate as a creativity enhancement technique for Website requirements elicitation. Requirements Engineering Journal **17** (2012) 171–186
3. Sakhnini, V., Mich, L., Berry, D.M.: On the sizes of groups using the full and optimized EPMcreate creativity enhancement technique for Web site requirements elicitation. In: Proceedings of the Workshop on Creativity in Requirements Engineering (CreaRE) at REFSQ'2013. (2013)

To see these figures better, zoom in on an electronic copy.

$V1$	$V2$	$f0$	$f1$	$f2$	$f3$	$f4$	$f5$	$f6$	$f7$	$f8$	$f9$	$f10$	$f11$	$f12$	$f13$	$f14$	$f15$
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Fig. 1. Table of the 16 Combinations of Two Viewpoints: "Vn" means "Stakeholder n's Viewpoint" and "fi" means "boolean function i"

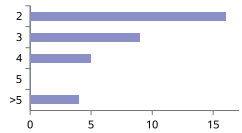

Fig. 6. Sizes of Ideal Groups of Business or Requirements Analysts

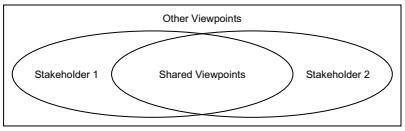

Fig. 2. Venn Diagram of Two Stakeholders' Viewpoints

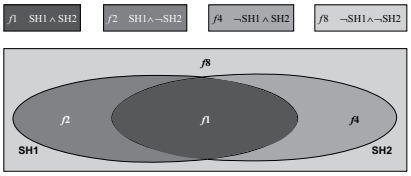

Fig. 3. The Four Steps of the Optimization and the Four Regions of the Venn Diagram

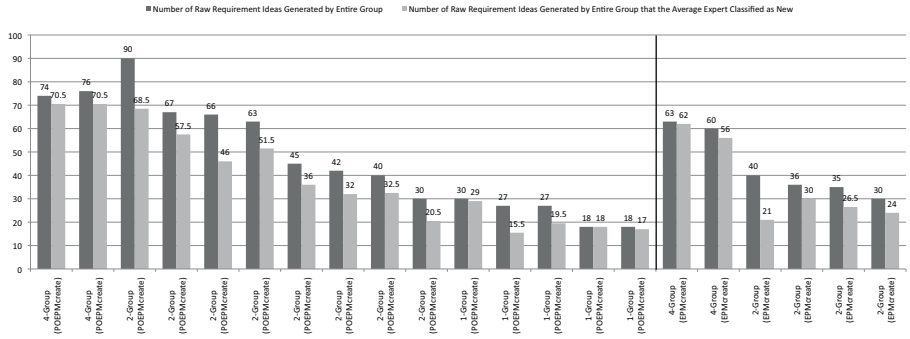

Fig. 4. Number of Raw and New Requirements Ideas Generated by Entire Groups

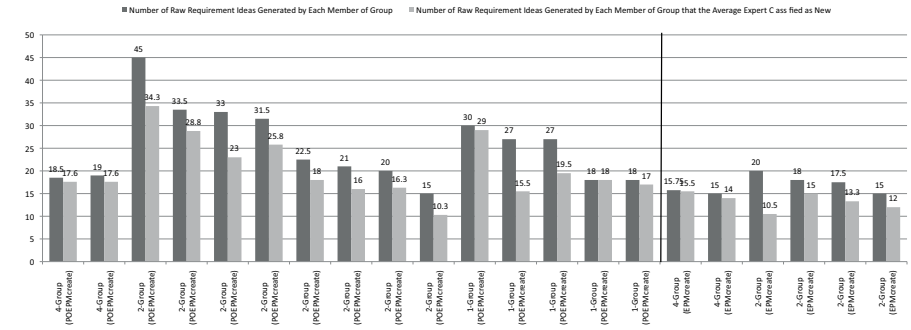

Fig. 5. Number of Raw and New Requirements Ideas Generated by Each Member of Groups

Security requirements analysis based on security and domain ontologies

Amina Souag[1], Camille Salinesi[1], Isabelle Wattiau[2]
[1] CRI, Paris 1 Sorbonne University
{Amina.Souag, camille.salinesi}@malix.univ paris1.fr
[2] CEDRIC CNAM & ESSEC Business School, France
isabelle.wattiau@cnam.fr

Security is the discipline concerned with protecting systems from a wide range of threats (malice, error or mischief) that break the system by exploiting a vulnerability, i.e. a property of the system or its environment that, when faced with particular threats, can lead to failure[5] . Security is a multi-faceted problem; it is as much about understanding the domain in which systems operate as it is about the systems themselves. While developing security facilities such as encryption, identity control, or specific architectures is important, our attention should be drawn at looking into the sociotechnical context in which target systems will operate and threats that may arise and their potential harm, so as to uncover security requirements. Recent research has argued about the importance of considering security at the early stages of the information systems development process, and especially the need to consider security during RE.

An ontology, in the field of knowledge representation, is most often defined as "a representation of a conceptualization"[1]. It should represent a shared conceptualization in order to have any useful purpose [2]. Ontologies are useful for representing and interrelating many types of knowledge. Several security ontologies have been proposed [3]. Domain ontologies are formal descriptions of classes of concepts and relationships between these concepts that describe a given domain.

Our previous experience with RITA [4] a requirements elicitation method that exploits a just one threat ontology, was that "being generic, the threats in the RITA ontology are not specific to the target [bank] industry" (the case study was in the banking sector). Experts involved in the evaluation complained about "the lack of specificity of the types of threats to the industry sector and the problem domain at hand". The problem that remains open is therefore that we need to exploit both security knowledge and domain knowledge to guide the elicitation of domain-specific security requirements. Our research question is "how to combine the use of security ontologies and domain ontologies to guide requirements elicitation efficiently?"

This paper presents an ongoing research project that aims to develop a method that explores the use of security and domain ontologies for SRE. The approach is generic in the sense that different security ontologies and different domain ontologies can be used with it. However it is domain specific when it is applied in the sense that during its application only one domain ontology is used.

Our method guides the discovery of security requirements for a specific domain. This process handled by a series of heuristic production rules that, starting from high level security requirements, produce a security requirements specification. Figure 1 shows an overview of our method. There are two sub-sets of rules. The first set of rules handles domain-specific analysis. The second set of rules performs a security specific analysis. Each set of rules exploits different ontologies: respectively domain ontologies and security ontologies. In order to be able to handle different security and domain ontologies, the rules were specified with so-called “upper ontologies”, that handle concepts that are (a) common to most ontologies, (b) sufficiently high level to abstract many other concepts in the specific ontologies, and (c) more importantly that represent an important subject of interest for the method.

The requirements definition process starts with the elicitation step, where stakeholders express their needs about security in non-formal sentences. Then an analysis stage is carried out to discover more requirements and express these needs in a semi-formal requirement.

During the elicitation step, an initial I* requirements model is first constructed from the stakeholders' needs and concerns expressed about security at the beginning of the project. At this stage, the analyst defines initial actors, resources, and especially security goals (integrity, confidentiality, traceability...) During the security requirements analysis stage, the production rules will exploit the security-specific ontology to discover threats, vulnerabilities, countermeasures, and resources, and thus enrich the requirements model by adding new elements (malicious tasks, vulnerability points...). During the domain specific security requirements analysis stage, another set of rules explores the domain ontology to improve the requirements model with resources, actors and other concepts that are more specific to the domain at hand; for instance: thieves in the banking domain, hijackers in the aeronautic domain, pirates in the maritime domain, etc.

The originality of the method lies: (a) in the fact that the combination of security and domain ontologies is not achieved a priori, but at runtime, while the method is applied, and (b) in the genericity of the method, in the sense that it is designed to be used with any pair of security and domain ontologies, as long as they embed some expected knowledge.

Our preliminary evaluation conducted through a small, but real, case study and through critical analysis by three experts (domain, security, requirements engineering, respectively). The evaluation shows that the method provides a good balance between the genericity with respect to the ontologies (which do not need to be selected in advance), and the specificity of the elicited requirements with respect to the domain at hand.

References

[1] T. R. Gruber, « Toward principles for the design of ontologies used for knowledge sharing? », Int. J. Hum.-Comput. Stud., vol. 43, nº 5‑6, p. 907‑928, nov. 1995.

[2] G. Dobson et P. Sawyer, Revisiting Ontology-Based Requirements Engineering in the age of the Semantic Web. In: Dependable Requirements Engineering of Computerised Systems at NPPs. 2006.
[3] S. Fenz et A. Ekelhart, « Formalizing information security knowledge », in Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, New York, NY, USA, 2009, p. 183–194.
[4] C. Salinesi, E. Ivankina, et W. Angole, « Using the RITA Threats Ontology to Guide Requirements Elicitation: an Empirical Experiment in the Banking Sector », in Managing Requirements Knowledge, 2008. MARK '08. First International Workshop on, 2008, p. 11 ‑ 15.
[5] R. J. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems. John Wiley & Sons, 2010.

The Recommendation in the Product Line Configuration Process

Raouia TRIKI , Camille SALINESI

CRI, University of Paris1 Panthéon Sorbonne, Paris, France
{Raouia.Triki, Camille.Salinesi}@univ-paris1.fr

Keywords: Product Line, Configuration, Recommendation, Requirement

In Product Line Engineering (PLE), product configuration describes the process of developing a product according to user requirements, by reuse from a Product Line model (PLM). The problem is that there are so many products in a PL that it is impossible to specify all of them explicitly. Then, when a user makes a decision (e.g. require or reject a reusable artifact), this can be contradictory with former decisions. Thus, the user will be confused and she/he will eventually abandon the configuration process. Consequently, it's crucial to guide the user by combining recommendation and configuration in the PL configuration process.

Recommendation helps users to identify relevant products according to their requirements which are elicited by observing purchase habits, features of products formerly acquired, etc. Several recommendation techniques already exist [1]. Many of these techniques are made for simple products and are not adapted to complex systems such as product lines or configurable software.

Our research goal is to help user to make his choice in a dynamic way by combining recommendation and configuration. This aim to inform the user in real time about possible/unattainable features according to her/his choices, and to suggest decision by reasoning with known configurations.

Thus, the main objective of our research goal is to answer the following research question:

How to combine the recommendation and the configuration in a product line?

In addition, another problem arises: at which configuration process level the recommendation should be applied?

In order to achieve our research goal, we are led to study and experiment the different recommendation techniques by exploring the limits and interests of each one. This study allows adapting these techniques to the PL configuration process.

The idea of our approach is to recommend only partial configurations that are valid with respect to the PL constraints, and satisfy the requirements that the user has already specified.

Our approach consists in intertwining recommendation and configuration activities in an iterative way. At each iteration (i) a series of decisions is offered to the user, (ii) the user makes choices, (iii) testing the user partial configuration, (iv) recommendation (v) configuration and constraint propagation, (vi) final decision.

At the beginning of our research, our work was started with an "a priori configuration approach" by focusing on a pack of features. This is called a "partial configuration". After checking that the partial configuration is correct and if the number of candidate products is too high for manual selection, then recommendation can be used to help the user make decisions for the next pack of features. The recommendation for the next pack of features is done under the form of a list of partial configurations ordered from the most recommended to the least recommended. Then, the user selects from the list according to her/his requirements. Next, the process is repeated at each cycle until the user decides to stop.

There are several recommendation techniques that can be applied. Among these techniques, we have started our research work by handling the content based recommendation method [5] which is based on textual data and it treats the recommendation problem as a search for related items [1]. Content based recommendation technique uses the definition of existing products, which are defined as a combination of features, to support recommendation.

In the literature, there are several recommendation techniques. We distinguish techniques that are used for simple product catalogs. The "content based" filtering [5] makes recommendations from products that the user has chosen, while the "collaborative" filtering [3] makes recommendations from products that were purchased from other users.

On the other hand, there is recommendation techniques used for complex products like the "constraint based" recommendation [2]. It is defined as a constraint satisfaction problem (CSP) [4]. This recommendation technique provides a solution which is consistent with the PL constraints and satisfying the user requirements. Constraint based recommenders provide explanations for inconsistent requirements [2] such that the calculation of recommendation becomes possible.

The major goal for future work is to experiment the different recommendation techniques and adapt them to our problem. In addition, we intend to extend our approach by defining the level of recommendation. Indeed, the recommendation should be applied, first, on the order of features among which the user selects a partial configuration. Secondly, the recommendation should be applied on the features selection as has been shown in our approach.

References

1. Balabanovic, M., Shoham, Y.: Combining Content-based and Collaborative Recommendation. In: Communications of the ACM, 40 (3), (1997)
2. Felfernig, A., Burke, R.: Constraint-based Recommender Systems: Technologies and Research Issues. In: Proceedings of the ACM International Conference on Electronics Commerce (ICEC'08), pp. 17--26, Innsbruck, Austria (2008)
3. Linden, G., Smith, B., York, J.: Amazon.com Recommendations: item-to-item collaborative filtering. In: IEEE Internet Computing, pp. 76--80 (2003)
4. Tsang, E.: Foundations of Constraint Satisfaction. Academic Press, London and San Diego (1993)
5. Van Meteren, R., Van Someren, M.: Using Content-based Filtering for Recommendation. In: ECMIL 2000 workshop (2000)

Previously published ICB - Research Reports

2013

No 55 (May)

Daun, Marian; Fockel, Markus; Holtmann, Jörg; Tenbergen, Bastian: "Goal-Scenario-Oriented Requirements Engineering for Functional Decomposition with Bidirectional Transformation to Controlled Natural Language. Case Study "Body Control Module"

No 54 (March)

Fischotter, Melanie; Goedicke, Michael; Kurz-Karaoglu, Filiz; Schwinning, Nils; Striewe, Michael: "Erster Jahresbericht zum Projekt „Bildungsgerechtigkeit im Fokus" (Teilprojekt 1.2 – „Blended Learning") an der Fakultät für Wirtschaftswissenschaften"

2012

No 53 (December)

Frank, Ulrich: "Thoughts on Classification / Instantiation and Generalisation / Specialisation"

No 52 (July)

Berntsson-Svensson, Richard; Berry, Daniel; Daneva, Maya; Dörr, Jörg; Fricker, Samuel A.; Hermann, Andrea; Herzwurm, Georg; Kauppinen, Marjo; Madhayji, Nazim H.; Mahaux, Martin; Paech, Barbara; Penzenstadler, Birgit; Pietsch, Wolfram; Salinesi, Camille; Schneider, Kurt; Seyff, Norbert; van de Weerd, Inge: "18th International Working Conference on Requirements Engineering: Foundation for Software Quality. Proceedings of the Workshops RE4SuSy, REEW, CreaRE, RePriCo, IWSPM and the Conference Related Empirical Study, Empirical Fair and Doctoral Symposium"

No 51 (May)

Frank, Ulrich: "Specialisation in Business Process Modelling: Motivation, Approaches and Limitations"

No 50 (March)

Adelsberger, Heimo; Drechsler, Andreas; Herzig, Eric; Michaelis, Alexander; Schulz, Philip; Schütz, Stefan; Ulrich, Udo: "Qualitative und quantitative Analyse von SOA-Studien. Eine Metastudie zu serviceorientierten Architekturen"

2011

No 49 (December)

Frank, Ulrich: "MEMO Organisation Modelling Language (2): Focus on Business Processes"

No 48 (December)

Frank, Ulrich: "MEMO Organisation Modelling Language (1): Focus on Organisational Structure"

No 47 (December)

Frank, Ulrich: "MEMO Organisation Modelling Language: Requirements and Core Diagram Types"

No 46 (December)

Frank, Ulrich: "Multi-Perspective Enterprise Modelling: Background and Terminological Foundation"

No 45 (November)

Frank, Ulrich; Strecker, Stefan; Heise, David; Kattenstroth, Heiko; Schauer, Carola: "Leitfaden zur Erstellung wissenschaftlicher Arbeiten in der Wirtschaftsinformatik"

No 44 (September)

Berenbach, Brian; Daneva, Maya; Dörr, Jörg; Fricker, Samuel; Gervasi, Vincenzo; Glinz, Martin; Hermann, Andrea; Krams; Benedikt; Madhavji, Nazim H.; Paech, Barbara; Schockert, Sixten; Seyff, Norbert (Eds): "17th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2011). Proceedings of the REFSQ 2011 Workshops REEW, EPICAL and RePriCo, the REFSQ 2011 Empirical Track (Empirical Live Experiment and Empirical Research Fair), and the REFSQ 2011 Doctoral Symposium"

No 43 (February)

Frank, Ulrich: "The MEMO Meta Modelling Language (MML) and Lnguage Architecture – 2nd Edition"

2010

No 42 (December)

Frank, Ulrich: "Outline of a Method for Designing Domain-Specific Modelling Languages"

No 41 (December)

Adelsberger,Heimo; Drechsler, Andreas (Eds): "Ausgewählte Aspekte des Cloud-Computing aus einer IT-Management-Perspektive – Cloud Governance, Cloud Security und Einsatz von Cloud Computing in jungen Unternehmen"

No 40 (October 2010)
Bürsner, Simone; Dörr, Jörg; Gehlert, Andreas; Herrmann, Andrea; Herzwurm, Georg; Janzen, Dirk; Merten, Thorsten; Pietsch, Wolfram; Schmid, Klaus; Schneider, Kurt; Thurimella, Anil Kumar (Eds): "16th International Working Conference on Requirements Engineering: Foundation for Software Quality. Proceedings oft he Workshops CreaRE, PLREQ, RePriCo and RESC"

No 39 (May 2010)
Strecker, Stefan; Heise, David; Frank, Ulrich: "Entwurf einer Mentoring-Konzeption für den Studiengang M.Sc. Wirtschaftsinformatik an der Fakultät für Wirtschaftswissenschaften der Universität Duisburg-Essen"

No 38 (February 2010)
Schauer, Carola: "Wie praxisorientiert ist die Wirtschaftsinformatik? Einschätzungen von CIOs und WI-Professoren"

No 37 (January 2010)
Benavides, David; Batory, Don; Grunbacher, Paul (Eds.): "Fourth International Workshop on Variability Modelling of Software-intensive Systems"

2009

No 36 (December 2009)
Strecker, Stefan: "Ein Kommentar zur Diskussion um Begriff und Verständnis der IT-Governance - Anregungen zu einer kritischen Reflexion"

No 35 (August 2009)
Rüngeler, Irene; Tüxen, Michael; Rathgeb, Erwin P.:"Considerations on Handling Link Errors in STCP"

No 34 (June 2009)
Karastoyanova, Dimka; Kazhamiakan, Raman; Metzger, Andreas; Pistore, Marco (Eds.): "Workshop on Service Monitoring, Adaption and Beyond"

No 33 (May 2009)
Adelsberger,Heimo; Drechsler , Andreas; Bruckmann, Tobias; Kalvelage, Peter; Kinne, Sophia; Pellinger, Jan; Rosenberger, Marcel; Trepper, Tobias: „Einsatz von Social Software in Unternehmen – Studie über Umfang und Zweck der Nutzung"

No 32 (April 2009)
Barth, Manfred; Gadatsch, Andreas; Kütz, Martin; Rüding, Otto; Schauer, Hanno; Strecker, Stefan: „Leitbild IT-Controller/-in – Beitrag der Fachgruppe IT-Controlling der Gesellschaft für Informatik e. V."

No 31 (April 2009)
Frank, Ulrich; Strecker, Stefan: "Beyond ERP Systems: An Outline of Self-Referential Enterprise Systems – Requirements, Conceptual Foundation and Design Options"

No 30 (February 2009)
Schauer, Hanno; Wolff, Frank: „Kriterien guter Wissensarbeit – Ein Vorschlag aus dem Blickwinkel der Wissenschaftstheorie (Langfassung)"

No 29 (January 2009)
Benavides, David; Metzger, Andreas; Eisenecker, Ulrich (Eds.): "Third International Workshop on Variability Modelling of Software-intensive Systems"

2008

No 28 (December 2008)
Goedicke, Michael; Striewe, Michael; Balz, Moritz: „Computer Aided Assessments and Programming Exercises with JACK"

No 27 (December 2008)
Schauer, Carola: "Größe und Ausrichtung der Disziplin Wirtschaftsinformatik an Universitäten im deutschsprachigen Raum - Aktueller Status und Entwicklung seit 1992"

No 26 (September 2008)
Milen, Tilev; Bruno Müller-Clostermann: " CapSys: A Tool for Macroscopic Capacity Planning"

No 25 (August 2008)
Eicker, Stefan; Spies, Thorsten; Tschersich, Markus: "Einsatz von Multi-Touch beim Softwaredesign am Beispiel der CRC Card-Methode"

No 24 (August 2008)
Frank, Ulrich: "The MEMO Meta Modelling Language (MML) and Language Architecture – Revised Version"

No 23 (January 2008)
Sprenger, Jonas; Jung, Jürgen: "Enterprise Modelling in the Context of Manufacturing – Outline of an Approach Supporting Production Planning"

No 22 (January 2008)
Heymans, Patrick; Kang, Kyo-Chul; Metzger, Andreas, Pohl, Klaus (Eds.): "Second International Workshop on Variability Modelling of Software-intensive Systems"

2007

No 21 (September 2007)
Eicker, Stefan; Annett Nagel; Peter M. Schuler: "Flexibilität im Geschäftsprozess-management-Kreislauf"

No 20 (August 2007)
Blau, Holger; Eicker, Stefan; Spies, Thorsten: "Reifegradüberwachung von Software"

No 19 (June 2007)
Schauer, Carola: "Relevance and Success of IS Teaching and Research: An Analysis of the ‚Relevance Debate'

No 18 (May 2007)
Schauer, Carola: "Rekonstruktion der historischen Entwicklung der Wirtschaftsinformatik: Schritte der Institutionalisierung, Diskussion zum Status, Rahmenempfehlungen für die Lehre"

No 17 (May 2007)
Schauer, Carola; Schmeing, Tobias: "Development of IS Teaching in North-America: An Analysis of Model Curricula"

No 16 (May 2007)
Müller-Clostermann, Bruno; Tilev, Milen: "Using G/G/m-Models for Multi-Server and Mainframe Capacity Planning"

No 15 (April 2007)
Heise, David; Schauer, Carola; Strecker, Stefan: "Informationsquellen für IT-Professionals – Analyse und Bewertung der Fachpresse aus Sicht der Wirtschaftsinformatik"

No 14 (March 2007)
Eicker, Stefan; Hegmanns, Christian; Malich, Stefan: "Auswahl von Bewertungsmethoden für Softwarearchitekturen"

No 13 (February 2007)
Eicker, Stefan; Spies, Thorsten; Kahl, Christian: "Softwarevisualisierung im Kontext serviceorientierter Architekturen"

No 12 (February 2007)
Brenner, Freimut: "Cumulative Measures of Absorbing Joint Markov Chains and an Application to Markovian Process Algebras"

No 11 (February 2007)
Kirchner, Lutz: "Entwurf einer Modellierungssprache zur Unterstützung der Aufgaben des IT-Managements – Grundlagen, Anforderungen und Metamodell"

No 10 (February 2007)
Schauer, Carola; Strecker, Stefan: "Vergleichende Literaturstudie aktueller einführender Lehrbücher der Wirtschaftsinformatik: Bezugsrahmen und Auswertung"

No 9 (February 2007)
Strecker, Stefan; Kuckertz, Andreas; Pawlowski, Jan M.: "Überlegungen zur Qualifizierung des wissenschaftlichen Nachwuchses: Ein Diskussionsbeitrag zur (kumulativen) Habilitation"

No 8 (February 2007)
Frank, Ulrich; Strecker, Stefan; Koch, Stefan: "Open Model - Ein Vorschlag für ein Forschungsprogramm der Wirtschaftsinformatik (Langfassung)"

2006

No 7 (December 2006)
Frank, Ulrich: "Towards a Pluralistic Conception of Research Methods in Information Systems Research"

No 6 (April 2006)
Frank, Ulrich: "Evaluation von Forschung und Lehre an Universitäten – Ein Diskussionsbeitrag"

No 5 (April 2006)
Jung, Jürgen: "Supply Chains in the Context of Resource Modelling"

No 4 (February 2006)
Lange, Carola: "Development and status of the Information Systems / Wirtschaftsinformatik discipline: An interpretive evaluation of interviews with renowned researchers, Part III – Results Wirtschaftsinformatik Discipline"

2005

No 3 (December 2005)
Lange, Carola: "Development and status of the Information Systems / Wirtschaftsinformatik discipline: An interpretive evaluation of interviews with renowned researchers, Part II – Results Information Systems Discipline"

No 2 (December 2005)
Lange, Carola: "Development and status of the Information Systems / Wirtschaftsinformatik discipline: An interpretive evaluation of interviews with renowned researchers, Part I – Research Objectives and Method"

No 1 (August 2005)
Lange, Carola: „Ein Bezugsrahmen zur Beschreibung von Forschungsgegenständen und -methoden in Wirtschaftsinformatik und Information Systems"

Research Group	Core Research Topics
Prof. Dr. H. H. Adelsberger Information Systems for Production and Operations Management	E-Learning, Knowledge Management, Skill-Management, Simulation, Artificial Intelligence
Prof. Dr. F. Ahlemann Information Systems and Strategic Management	Strategic planning of IS, Enterprise Architecture Management, IT Vendor Management, Project Portfolio Management, IT Governance, Strategic IT Benchmarking
Prof. Dr. P. Chamoni MIS and Management Science / Operations Research	Information Systems and Operations Research, Business Intelligence, Data Warehousing
Prof. Dr. K. Echtle Dependability of Computing Systems	Dependability of Computing Systems
Prof. Dr. S. Eicker Information Systems and Software Engineering	Process Models, Software-Architectures
Prof. Dr. U. Frank Information Systems and Enterprise Modelling	Enterprise Modelling, Enterprise Application Integration, IT Management, Knowledge Management
Prof. Dr. M. Goedicke Specification of Software Systems	Distributed Systems, Software Components, CSCW
Prof. Dr. V. Gruhn Software Engineering	Design of Software Processes, Software Architecture, Usability, Mobile Applications, Component-based and Generative Software Development
PD Dr. C. Klüver Computer Based Analysis of Social Complexity	Soft Computing, Modeling of Social, Cognitive, and Economic Processes, Development of Algorithms
Prof. Dr. T. Kollmann E-Business and E-Entrepreneurship	E-Business and Information Management, E-Entrepreneurship/E-Venture, Virtual Marketplaces and Mobile Commerce, Online-Marketing
Prof. Dr. K. Pohl Software Systems Engineering	Requirements Engineering, Software Quality Assurance, Software-Architectures, Evaluation of COTS/Open Source-Components
Prof. Dr. Ing. E. Rathgeb Computer Network Technology	Computer Network Technology
Prof. Dr. R. Unland Data Management Systems and Knowledge Representation	Data Management, Artificial Intelligence, Software Engineering, Internet Based Teaching
Prof. Dr. S. Zelewski Institute of Production and Industrial Information Management	Industrial Business Processes, Innovation Management, Information Management, Economic Analyses

ISSN 1860-2770 (Print)
ISSN 1866-5101 (Online)