18th International working conference on requirements engineering: foundation for software quality

Richard Berntsson Svensson, Daniel Berry, Maya Daneva, Jörg Dörr, Samuel A. Fricker, Andrea Herrmann, Georg Herzwurm, Marjo Kauppinen, Nazim H. Madhavji, Martin Mahaux, Barbara Paech, Birgit Penzenstadler, Wolfram Pietsch, Camille Salinesi, Kurt Schneider, Norbert Seyff, Inge van de Weerd (Eds.)

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

ICB

Institut für Informatik und Wirtschaftsinformatik

52

ICB-RESEARCH REPORT

UNIVERSITÄT DUISBURG ESSEN

Proceedings Edited by:

Richard Berntsson Svensson

Daniel Berry

Maya Daneva

Jörg Dörr

Samuel A. Fricker

Andrea Herrmann

Georg Herzwurm

Marjo Kauppinen

Nazim H. Madhavji

Martin Mahaux

Barbara Paech

Birgit Penzenstadler

Wolfram Pietsch

Camille Salinesi

Kurt Schneider

Norbert Seyff

Inge van de Weerd

ICB Research Reports

Edited by:

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

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 2012 in Essen, Germany.

- Engineering for Sustainable Systems (RE4SuSy)
- Requirements Engineering Efficiency Workshop (REEW 2012)
- Creativity in Requirements Engineering (CreaRE 2012)
- Requirements Prioritization for customer oriented Software Development (RePriCo)
- International Workshop on Software Product Management (IWSPM)
- Alive Empirical Study
- Online Questionnaires
- Empirical Research Fair
- Doctoral Symposium

Table of Contents

Part I

REFSQ 2012 Workshop Proceedings

1 Preface

Editor

Samuel A. Fricker
Blekinge Institute of Technology, Sweden, samuel.fricker@bth.se

Preface from the RefsQ 2012 Workshops Chair

Samuel A. Fricker

Blekinge Institute of Technology, School of Computing
Campus Gräsvik, 371 79 Karlskrona, Sweden
samuel.fricker@bth.se

Conference workshops are important forum to initiate new research and to develop young researchers. This is especially true for the *International Working Conference on Requirements Engineering: Foundation for Software Quality (RefsQ)* series, which targets an "I heard it first at RefsQ!" experience. The RefsQ workshops allow researchers to expose their research ideas and early results. Each workshop provides time and an interested audience from industry and academia to discuss the presented ideas. In addition, the RefsQ workshops allow young, promising researchers to plan and implement a researcher meeting for the first time. This experience and the network they develop enable them to actively participate in the research community.

RefsQ 2012 called for proposals of workshops that have the potential to significantly advance requirements engineering. Such workshops cover topics that are important for practice, are new to the field, have controversial viewpoints, and are unsatisfactorily understood. The dialogue among participants shall lead to interesting follow-up research, empirical investigations, and industrial practice improvement.

The workshop proposals were evaluated based on the following criteria. A workshop should be led by a senior and a junior researcher to transfer knowledge and research culture. Its topic should be novel to enable growth of the field. It should attract both earlier and new RefsQ participants to enable growth of the community. Its format should allow generating, rather than only consuming knowledge. Finally, to enable innovation, established workshops were only accepted if successful previously.

RefsQ 2012 accepted five workshops. The new *International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy)* addressed requirements engineering in the sustainability context, which has become important for our society. The *Requirements Engineering Efficiency Workshop (REEW)* was held for the second time to discuss approaches for increasing requirements engineering efficiency. The workshop on *Creativity in Requirements Engineering (CREARE)* was held for the second time to address requirements engineering in an innovation context. The workshop on *Requirements Prioritization for Customer Oriented Software Development (RePriCo)* was held for the third time to discuss prioritization of requirements. The *International Workshop on Software Product Management (IWSPM)* joined RefsQ for the first time to discuss approaches for managing software as a product. This proceedings explains the paper selection processes and includes the accepted contributions.

On behalf of the RefsQ organization committee, I would like to thank all workshop organizers and contributors to their excellent work. The workshops fulfilled their expectations to our highest satisfaction.

2 Requirements Engineering for Sustainable Systems (RE4SuSy)

Editors

Birgit Penzenstadler
Technische Universittät München, Germany, penzenst@in.tum.de

Martin Mahaux
University of Namur, Belgium, martin.mahaux@fundp.ac.be

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

Workshop Programme

First International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy)

Birgit Penzenstadler (Organization Chair), Martin Mahaux (Organization Chair), and Camille Salinesi (Program Chair)

[1] Technische Universität München, Germany, `penzenst@in.tum.de`
[2] University of Namur, Belgium, `martin.mahaux@fundp.ac.be`
[3] Université Paris 1 - Sorbonne, France, `camille@univ-paris1.fr`

Abstract. Researchers have recently started to explore how to support the elicitation and documentation of sustainability requirements. In the mean time, ubiquitous socio-technical systems alter the way we live, and consequently have a potentially huge impact on sustainability. As sustainability is one of the biggest challenges facing humanity in the coming decades, we must reinforce research in this direction and ensure it is appropriately rooted in the practice. The workshop provided an interactive stage to collaboratively define a research agenda in RE for sustainable systems, and also to jumpstart collaboration through networking and active discussion on concrete points of this agenda.

Keywords: requirements, sustainability, environment, society

1 Background & Goals

ICT-based systems are tremendously affecting the way we interact with the world around us. These changes occur at a high rate and in shortening innovation cycles. As suggested by the Smart2020 report [1], ICT can play a positive role towards a more sustainable world. In that context, requirements engineers will be key in ensuring that not only present needs, but also future generations needs, can be satisfied. Indeed, in order to use the potential of ICT to reach more sustainable behaviors, sustainability should be made a first class quality requirement. This is our overarching goal: ensure that sustainability requirements are systematically and adequately elicited and documented when developing socio-technical systems.

2 Addressed Themes

The most cited definition of the term "sustainable development" stems from the so-called Brundtland report ("Our common future" [2]): "Sustainable development is development that meets the needs of the present without compromising the ability of future generations to meet their own needs". It is interesting to

note that, if it is commonly accepted that RE is mainly concerned with satisfying present needs, then "sustainable RE" is a natural extension to this understanding, anticipating on the satisfaction of future needs.

Sustainability has three major pillars: environment, society and economy. Economy being targeted by traditional RE, we will concentrate on the two others. Examples of environmental sustainability in RE research can be found in [3–5]. The november 2010 edition of the IEEE Computer journal [6], addressing Technology Mediated Social Participation gives an excellent idea of how ICT is related to social sustainability. Although not limited to these items, the workshop fosters discussion on:

- how requirements engineering can help in analysing sustainability issues;
- how to adapt existing or invent new elicitation, documentation, validation techniques and tools for sustainability requirements;
- how to model sustainability requirements with all necessary context;
- how to learn from and interact with other sustainability-related domains (e.g., environmental informatics);
- how to define, measure and assess sustainability as quality attribute.

As sustainability is a global and pervasive challenge, no particular industry sector is excluded from our analysis. Any human activity that has an impact on its society or its environment and involves a socio-technical system is on our focus. Our aim is to see how such a socio-technical system can be better designed to reduce its negative impacts, and strengthen the positive ones. However, some industry sectors have been particularly under focus for the envisioned improvement. The smart2020 report [1], Van Ypersele's keynote at RE'08 conference [7] and Pirolli et al. [6] suggest fields like Energy Supply, Transports, Buildings, Agriculture, Waste, Governance, Health and more.

3 Submissions and Selection Process

In order to reach the goals of the workshop, we encouraged short submissions formats for Problem Statements, Visions, Research Preview, Ongoing Research Projects, Research Results. We invited posters, video clips or multi-media presentations of up to seven minutes with a one page abstract. We also invited short papers of up to 6 pages LNCS style if authors wish to submit a more polished relevant research.

For the selection process, the Program Chair assigned each submission to three members of the Program Committee (PC) for a formal blind review process. All authors (including the two Organization co-Chairs) indicated their Conflicts of Interests with the PC members, so reviews could be performed adequately. The PC members were Lorenz Hilty (University of Zürich), Steffen Zschaler (King's College London), Ruzanna Chitchyan (Leicester University), Stefan Naumann (Trier University of Applied Sciences), Bill Tomlinson (University of California, Irvine), Toni Ahlqvist (VTT Finland), Brian Donnellan (University of Ireland, Maynooth), David Stefan (University College London),

Emmanuel Letier (University College London), Andrea Zisman (City University London), Debra Richardson (University of California, Irvine), and Alistair Mavin (Rolls Royce, UK).

Being a starting community, and given the workshop's goals, we asked the PC members to focus their review on the relevance for the workshop and the potential for triggering discussion on a research agenda for RE4SuSy, rather than on maturity of the work or strength of the validation.

The reviews were published on the workshop wiki (`https://sustainability.wiki.tum.de/RE4SuSy`) along with the papers to kickstart the discussion process between all the stakeholders. The goal was to have authors enhancing their papers guided by the reviews and the potential comments from other workshop participants. This also made the review process entirely transparent. All submitted contributions were finally accepted. While this rate can be interpreted as a sign of looseness of the review process, we regard it as an effect of the positive and constructive review process and the quality of initial submissions.

4   Workshop Format

The focus was on interaction and participation. After a short energizing exercise and peronal presentation, the authors had five minutes to present their contribution. These were followed by heavy discussions (up to 25 minutes), kickstarted by the discussant assigned to each paper. After the break we brainstormed about possible research agenda items for RE4SuSy. This resulted in a list of interesting topics for our community to work on. Below we summarize initial contributions and present those results.

5   Summary of Contributions

The submissions covered a vast area of expertise, indicating the breadth of the RE4SuSy topic. Mahaux and Canon suggested in a position paper that the concept of sustainability was indeed more complex than one could initially imagine, and that it's integration into RE would be even more complex. As a first answer to this problem, researchers are developing new RE approaches, frameworks and tools. Penzenstadler et al. described their plans towards a new RE approach tailored to SuSy. Kern et al. presented a multi-media poster for GREENSOFT, a conceptual reference model for Green and Sustainable Software. It tries to characterize the what, where, when, how and who of this topic. Hoesch-Klohe and Ghose suggested to use scenarios as a basis for analyzing environmentally aware systems, showing their amenability for identifying the (approximated) environmental performance of a system. Two contributions highlighted aspects of RE4SuSy in specific sectors, with more in details. Jacquemin and Mahaux presented their view on RE for smart grids and electro-mobility, while Deprez et al. presented challenges on energy and eco-aware RE for cloud applications.

6   Results

The raw brainstormings results are available online at `https://sustainability.wiki.tum.de/Research+Agenda+Items`.

They served as a basis for suggesting the following research directions:

1. Understanding sustainability and sustainable systems: building interdisciplinary platforms for undertaking RE4SuSy research. How can we understand what sustainability means and harness the knowledge of other disciplines to achieve sustainable systems, taking into account that there is no single definition for sustainability, as it depends at least on the context and evolve over time?
2. Roles and Scoping:
   – Is RE4SuSy different to ordinary RE? Or is it just another NFR to optimize?
   – Who are the main RE4SuSy stakeholders?
3. Vertical / illustrative case study (E-mobility, SOA, etc.). It is suggested that, in parallel to more theoretical studies, applied research on specific cases should be undertaken to get a feeling from the practice and test preliminary ideas. Specific interesting areas are suggested, such as Cloud Applications for 1st level impacts, and smart grids for 2nd level.
4. Quality models, metrics, impacts, attributes that will help characterize precisely sustainable systems.
5. Cross-disciplinary future road mapping. Ensuring the satisfaction of future needs requires having a look at the future. How can we impact the present by looking at the future?

For each of the topics, there were at least one or two workshop participants who wanted to actively conduct respective research.

7   Conclusion and Next Steps

The 1st International Workshop on Requirements Engineering for Sustainable Systems (RE4SuSy) was a success and we received a lot of positive feedback. We hope to organize the workshop next year, too, and to attract an increasing number of submissions and participants for advancing and promoting research on this challenging topic.

The wiki is still open so that workshop participants as well as further interested researchers and practitioners can discuss the topics of the research agenda. Our next steps are to establish the research collaborations that were initiated during the workshop. Thereby, the researcher who enlisted him-/herself for a specific item on the research agenda serves as leader for the collaboration on a designated topic and invites the others who were interested in contributing to that same research agenda item. All participants agreed that it was crucial to involve other disciplines and each of us is initiating contacts to researchers from disciplines also related to sustainability.

We are looking forward to prosperous collaborations that will provide a strong basis for a follow-up workshop.

References

1. The Climate Group: Smart 2020: Enabling the low carbon economy in the information age. Technical report, Global eSustainability Initiative (2008)
2. United Nations World Commission on Environment and Development: Our Common Future. In: United Nations Conference. (1987)
3. Mahaux, M., Heymans, P., Saval, G.: Discovering Sustainability Requirements: an Experience Report. In: 17th REFSQ. (2011)
4. Cabot, J., Easterbrook, S., Horkoff, J., Lessard, L., Liaskos, S., Mazon, J.N.: Integrating sustainability in Decision-Making processes: A modelling strategy. In: 31st ICSE. (2009)
5. Stefan, D., Letier, E., Barrett, M., Stella-Sawicki, M.: Goal-Oriented system modelling for managing environmental sustainability. In: Third Workshop on Software Research and Climate Change. (2011)
6. Pirolli, P., Preece, J., Shneiderman, B.: Cyberinfrastructure for social action on national priorities. IEEE Computer **43** (2010) 20–21
7. 16th IEEE International Requirements Engineering Conference. In: 16th IEEE International Requirements Engineering Conference. (2008)

Integrating Energy and Eco-Aware Requirements Engineering in the Development of Services-Based Applications on Virtual Clouds

Jean-Christophe Deprez, Ravi Ramdoyal, and Christophe Ponsard

CETIC - Center of Excellence in Information and Communication Technologies
29/3 Rue des Frères Wright, B-6041 Charleroi, Belgium
{jcd,rr,cp}@cetic.be - www.cetic.be

Abstract. Over the last decades, the energy and ecological footprint of ICT systems, in particular those hosted at data centers, has grown significantly and continues to increase at an exponential rate. In parallel, research in self-adaptation has yielded initial results where reconfiguration of ICT systems at runtime enables dynamic improved quality of service. However, little has been done with regards to requirement engineering for self-adaptive system for a lower energy and ecological footprint. This paper sketches a framework on how to best reconcile these aspects in a conscious way covering requirements, design and run-time, by capturing, reasoning, monitoring and acting upon a set of interlinked system goals. We highlight a number of important problems to overcome for the approach to be feasible, present our current view on it and state interesting research questions open for discussions.

Keywords: Energy and Eco-Aware Requirements, Services-Based Applications, Virtual Clouds

1 Introduction

In 2007, the total footprint of the ICT sector was already about 2% of the estimated total emissions resulting from human activities, and this amount is expected to exceed 6 % in 2020 [9]. In parallel, the Climate Savers Computing Initiative (CSCI, which involves Intel, IBM, and Google among others) main aim is to reduce annual CO2 emissions from the IT sector by 54 million metric tons by 2011 and an additional 38 million metric tons by 2015, which is the equivalent of € 3.75 billion in annual energy cost savings. Its next focus is on energy efficiency of computing equipment (including networking systems and devices), adoption and deployment of power management, and promotion of smart computing practices (particularly developers).

In response to this trend, hardware and software are designed to become more aware of their ecological impact. Among the current new trends, **cloud computing** has received considerable attention as a promising approach for delivering energy and eco-aware ICT services by improving the utilization of

data center resources. In principle, cloud computing can be an inherently energy-efficient technology for ICT provided that its potential for significant energy savings is fully achieved at operation time, for instance, by enabling an eco-aware management of a cloud infrastructure. Besides, a highly questionable assumption regarding energy-effectiveness is precisely that energy savings necessarily equate to reduce carbon emissions [14]. Virtualisation has increased the capability of self-adaptation and self-reconfiguration of systems transparent to the end users [5].

However current research results do not fully address the problem of energy and eco-awareness in virtualized cloud infrastructure:

– most of the research addresses design-time solutions to provide run-time adaptation, while requirements engineering for self-adaptive software systems has received less attention [16].
– as our dependency on such systems is increasing, the underlying energy costs are also rising, which stresses the need for new energy-efficient and eco-friendly technologies that enable new pricing models for data centers [3].
– the kind of energy source (green vs brown) is not taken into account.

Within this context, this paper introduces a new approach to help software engineers address energy and ecological requirements when developing service-based applications developed to run in virtualized cloud environment, as well as to produce self-adaptable architectures that can optimize the energy and ecological performance at runtime. This approach starts by promoting goal oriented requirements engineering (GORE), where energy goals will be elicited and refined into energy requirements that specify specific service level objectives (SLO) for the runtime behavior of the software service. Second, the approach guides software engineers in producing design models that can be self-adaptive to achieve energy performance at runtime while keeping other parameters of the quality of service under control.

The remainder of the paper is structured as follows. Section 2 first introduces the key concepts of the approach, which is presented in Section 3. Section 4 then highlight some related work. Section 5 finally summarises some key research questions.

2 A Goal-Oriented Background

In this section, we introduce key definitions and concepts used in the proposed approach, notably, goal oriented requirement engineering and measures and associated key performance indicators on energy and ecology in cloud environment.

Goal-oriented requirements engineering (GORE) relies on the use of goals for eliciting, elaborating, structuring, specifying, analyzing, negotiating, documenting, and modifying requirements [13]. Such use is based on a multi-view model showing how goals, objects, agents, scenarios, operations, and domain properties are inter-related in the system-as-is and the system-to-be. A goal is an intent that can address different types of aspects. For instance, a behavioral

goal describes how the expected system should behave, while a soft goal describes wishes with less clear-cut criteria (typically improve, increase/reduce or maximize/minimize a given property of the system). **Soft-goals are at the heart of the proposed approach, as they can deal with energy-effectiveness and eco-awareness notably through first, improved adaptability of the architecture of service-based applications and second, minimization of the associated energetic needs and ecological footprints of service-based applications in operation**. In GORE, Goals are refined in subgoals and other relationships between goals (such as obstacles, conflicts, reinforcement) are explicitly elicited to form a goal graph. Alternative designs can also be captured. A requirement is a terminal goal (lead node in a goal graph) which is under the responsibility of a single agent (human or sub-system). The satisfiability of a goal can be specified by a measurable key performance indicator (KPI).

In the proposed approach these goal constructs will be used to show explicitly how energy and ecological goals relate to other non-functional goals of the system-as-is or the system-to-be. We will also define energy and ecological key performances indicators.

In the context of cloud computing, the metrics used to measure KPIs on energy usually focus on the energy consumed by hardware in the data centers, which is however not the only dimension [1]. This raises the first question: **RQ#1: How to deal with the lack of normalization for energy-effectiveness metrics and the lack of ecological-awareness regarding available energy sources ?** Our idea is to overcome two of the main current shortcomings, namely the lack of normalization for energy-effectiveness metrics and the lack ecological-awareness regarding available energy sources. Energy normalization is important if new pricing models per energy consumption and carbon emission are to be developed by cloud infrastructure provider and perceived fair by service providers. In particular, pay per Watts could lead to different bills if the same service with same input is scheduled on older or new more efficient hardware. Green vs. brown energy measures also provides an important aspect to consider in pricing models. For instance, if a software service can easily be scheduled during green energy production peaks then it could be given priority in case of overbooking of service providers.

The collection of energy KPI is triggering a second research question: **RQ#2 How to match fine grained energy consumption of VMs and even software components in a VM with the limited capabilities of measurement at the hardware level only?** Indeed most data centers currently providing Infrastructure as a service (IaaS) are limited to general physical measures. A possible answer is that energy-consumption models have to be developed to normalize and estimate the desired measures as precisely as possible. For instance, the combination of CPU-usage percentage, disk accesses and network transfers measures will be used to define the energetic consumption of software services components. Kansal et al. have proposed a model to infer VM consumption from hardware energetic consumption [10] and could be explored to achieve finer grain measurements.

3 From Energy Requirements to Runtime Eco-driven Evolution

The scope targeted for the proposed approach is the following, on the one hand, the infrastructure (IaaS) provider owns the hardware and the virtual infrastructure software and on the other hand, the software (SaaS) service provider owns and packages a service-based application to be deployed and operated at the IaaS provider. In this setup, the SaaS provider has little control over the scheduling and placement policies of the IaaS provider. It is however anticipated that IaaS provider will publish the required KPI measurements. As mentioned in the definition section, IaaS providers only have measurements on hardware consumption at the server rack level; however, new accurate estimation models can help to infer energy measurement at the VM and soon at a finer grain software component in a VM. The proposed approach is independent of who provides the software specific energy measurements. It can be the IaaS provider or even an independent energy service provider who acts as a trusted third party between the IaaS and SaaS providers. The important aspect is that energy measurement be fair and trusted by the SaaS providers. The proposed approach also assumes that the IaaS provider accepts to share energy measurements with the SaaS provider who will in turn use these measurements to improve the quality profile of its software service-based application.

To reduce the energy-consumption and improve the eco-friendliness of a service-based application, we claim that energy and eco-awareness must become a core principle of the architecture, design and implementation of all software components involved at the different layers (Infrastructure and application). This rather disruptive, cleans slate approach, where different layers of an ICT system are re-designed and re-implemented to better handle a given concern, was followed with great success by Donofrio et al. [6] who showed how co-design with all aspects of the infrastructure and of the application in mind helps to make high power computing more efficient while consuming less energy.

Figure 1 gives a high level view of our approach. **At specification and design level, it starts with a requirements elicitation and analysis of a new software service partly driven by library of energy goals** explicitly related to other application?s functional and non-functional goals. This helps architects to select the most appropriate architecture for developing a self-adaptable software service, and second, to generate the KPI and thresholds specific to the software service under development. An interesting question is **RQ#3: how to relate KPI of contributing/conflicting goals?**. To some extend the normalization discussed earlier helps but multiple criteria must be taken into account to design system adaptation policies that balance ecological and other SLA goals appropriately.

The next step consists of propagating these KPI and thresholds at detailed design level, for instance, annotating elements of UML diagrams with particular energy KPI thresholds. These annotations are then used at compile time to inject the necessary measurement probes in the application to enable runtime measurements. These runtime measurements will then be used at three different
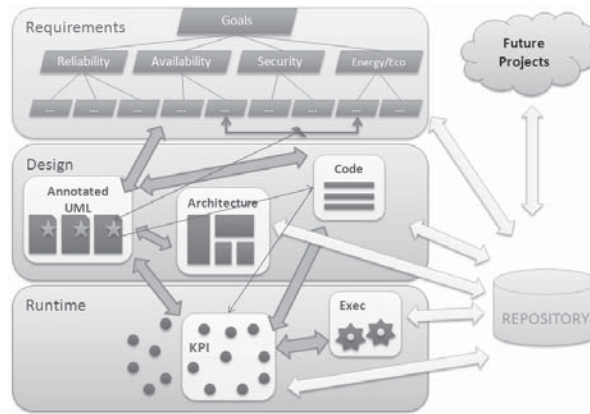
Fig. 1. Eco-aware Evolution Framework

levels, at software service operation level, at maintenance level of the particular software service and at a more general level for the development of new software services. The rest of this section details them.

At the service operation level, the KPI measurements are used by the service itself to perform self-adaptation actions that will improve its energy runtime performance while satisfying the other SLA aspects such as performance and security. Self-adaptation is limited to anticipated variability injected in the service architecture. A legitimate question is: **RQ#4: how to identify variability point at design time and design adaptations policies that balance ecological and other SLA goals**. For example, depending on the usage load, a self-adaptable system would vary its configuration between an energy costly mirror-oriented data storage and a more economic but also less available single centralized storage. In addition, an infrastructure is required to manage the KPI monitoring and adaptation policy rules. A question here is **RQ#5: which concrete and efficient form can this take in a SOA/Cloud architecture?** Middelware level will allow to benefit from application transparency and scalability but attention must be given to avoid consuming more energy than what is saved for example by triggering frequent reconfiguration or gathering too large amounts of historical data.

At the maintenance level, the KPI measurements provide valuable feedback to architects and developers of the measured software service. In turn, they can refactor the software service based on concrete energy data and clearly identify the energy bottlenecks of the software service. While self-adaptation can be performed along a few anticipated energy bottlenecks, the manual refactoring based on energy KPI will address more intricate behaviours of the software service that could not be anticipated at the design time.

At the general level, an overall guidance is needed to develop new service-based applications with better energy and ecological profiles. To formulate appropriate guidance to architects at requirement and design phase,

data on many applications are needed to cross relate their energy goals, their architectures, their variability points, etc. A question here is **RQ#6: What data on architectures, variability points to capture and cross-relate to KPI to enable efficient ecological guideance of future applications?**

4 Related Works

In practice, current research on energy-aware cloud computing is limited to improving the energy-efficient operation of computer hardware and network infrastructure. For instance, Intel has recently pushed server hardware with increased computing efficiency targeted for data center providing a virtual infrastructure [8], while [17, 11, 7] focused on the consolidation of virtualized infrastructure in data centers to improve energy efficiency. The FP7 research projects FIT4Green [2] and GAMES [4] are further advancing on consolidation techniques in virtualized environment, while [12] also proposes an approach to creating environmental awareness in service oriented software engineering.

However, none of these researches ensure energy-awareness at the different steps and levels of a service-based application to run in a virtualized cloud. In particular, very few methodology is currently proposed to support the requirements engineer and design modeling of systems that manages self-adaptation according to energy and eco-awareness. A good survey confirming the currently limited work devoted to this domain is presented in [15]. Without more energy consideration at the requirement and design phase, the development of energy-aware code at the various layers, infrastructure, middleware and service application is unlikely to be successful. We believe that the proposed approach that supports the requirements engineering and design modeling for energy-and eco-aware, self-adaptive systems will contribute further improve the energy and ecological profile of ICT systems running in virtualised cloud environments.

5 Conclusion and Future Works

In this paper, we sketched an approach to improve the ecological awareness of service-based applications. Our goal is not to propose a definitive solution but rather to highlight a number of open research questions and propose some partial answers. To increase the impact of the approach, it is worth noting that its application is not limited to new development project but is applicable to existing systems. The main difference resides in the self-adaptation, in particular, the architecture of an existing software service will not initially include well-defined and controlable variability points. Thus, the guidance on refactoring will also cover existing service-based systems.

References

1. Baliga, J., Ayre, R.W.A., Hinton, K., Tucker, R.S.: Green cloud computing: Balancing energy in processing, storage, and transport. In: Proceedings of the IEEE. vol. 99 (January 2011)

2. Basmadjian, R., Bunse, C., Georgiadou, V., Giuliani, G., Klingert, S., Lovasz, G., Majanen, M.: Fit4green - energy aware ict optimization policies. In: Proc. COST Action IC0804 on Energy Efficiency in Large Scale Distributed Systems (2010)

3. Berl, A., Gelenbe, E., Di Girolamo, M., Giuliani, G., De Meer, H., Dang, M.Q., Pentikousis, K.: Energy-efficient cloud computing. The Computer Journal 53(7), 1045–1051 (2009)

4. Bertoncini, M., Pernici, B., Salomie, I., Wesner, S.: Games: Green active management of energy in it service centres. In: CAiSE Forum 2010, Hammamet, Tunisia, June 7-9. pp. 238–252 (2010)

5. Cheng, B.H.C.e.a.: Software engineering for self-adaptive systems: A research roadmap. In: Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) Software Engineering for Self-Adaptive Systems. Lecture Notes in Computer Science, vol. 5525, pp. 1–26. Springer (2009)

6. Donofrio, D.e.a.: Energy-efficient computing for extreme-scale science. Computer 42, 62–71 (November 2009)

7. Garg, S.K., Yeo, C.S., Buyya, R.: Green cloud framework for improving carbon efficiency of clouds. In: Proc. of the 17th Int. Conf. on Parallel Processing - Volume Part I. pp. 491–502. Euro-Par'11, Springer-Verlag, Berlin, Heidelberg (2011)

8. Intel: Breakthrough security capabilities and energy-efficient performance for cloud computing infrastructures (2010), `http://software.intel.com/file/26765`

9. Juan-Carlos Lï£¡pez-Lï£¡pez, Giovanna Sissa, L.N.: Green ict: The information society's commitment for environmental sustainability. In: UPGRADE, vol. XII. Council of European Professional Informatics Societies (CEPIS) (October 2011)

10. Kansal, A., Zhao, F., Liu, J., Kothari, N., Bhattacharya, A.A.: Virtual machine power metering and provisioning. In: Hellerstein, J.M., Chaudhuri, S., Rosenblum, M. (eds.) SoCC. pp. 39–50. ACM (2010), `http://doi.acm.org/10.1145/1807128.1807136`

11. Kim, K.H., Beloglazov, A., Buyya, R.: Power-aware provisioning of virtual machines for real-time cloud services. Concurr. Comput. : Pract. Exper. 23, 1491–1505 (September 2011)

12. Lago, P., Jansen, T.: Creating environmental awareness in service oriented software engineering. In: Proc. s of the 2010 Int. Conf. on Service-oriented Computing. pp. 181–186. ICSOC'10, Springer-Verlag, Berlin, Heidelberg (2011)

13. van Lamsweerde, A.: Requirements engineering: from system goals to UML models to software specifications. John Wiley and Sons, Ltd. (2009)

14. Linthicum, D.: Beware: Cloud computing's green claims aren't always true. Infoworld (July 2011), `http://www.infoworld.com/d/cloud-computing/beware-cloud-computings-green-claims-arent-always-true-167984`

15. Mahaux, M., Heymans, P., Saval, G.: Discovering sustainability requirements: An experience report. In: Berry, D.M., Franch, X. (eds.) REFSQ. Lecture Notes in Computer Science, vol. 6606, pp. 19–33. Springer (2011)

16. Qureshi, N.A., Perini, A.: Engineering adaptive requirements. In: Proceedings of the 2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems. pp. 126–131. IEEE Computer Society, Washington, DC, USA (2009)

17. Srikantaiah, S., Kansal, A., Zhao, F.: Energy aware consolidation for cloud computing. In: Proceedings of the 2008 conference on Power aware computing and systems. pp. 10–10. HotPower'08, USENIX Association, Berkeley, CA, USA (2008)

Making use of scenarios for environmentally aware system design

Konstantin Hoesch-Klohe, Aditya Ghose

Decision Systems Lab (DSL),
School of Computer Science and Software Engineering,
University of Wollongong.

Abstract. This paper motivates the use of *scenarios* as a basis for environmentally aware system design, by showing their amenability for identifying the (approximated) environmental performance of an to-be system. In particular, we describe two complementary techniques for assessing and comparing the environmental performance of scenarios and how this can promote environmentally friendly decision making.

Keywords: Environmentally aware design, Requirements Engineering (RE), Scenarios, Resource modelling, Non-functional requirements (NFR).

1 Introduction

While much research attention has focused on developing alternative energy sources, automotive technologies or waste disposal techniques, we often ignore the fact that our behaviour (or that of a system) is a critical contributor to our environmental footprint. It is therefore crucial that we start to analyse existing- and to-be system behaviour and the intentions that give rationale to the former, in the context of our accumulated environmental debts. Requirements engineering (RE), supports the identification, analysis and specification of stakeholder intentions and their refinement to a concrete system design, which gives rise to the particular behaviour from its behaviour. We therefore believe that RE is the right starting point for nurturing the development of environmentally friendly systems (this has also been pointed out in e.g. [1]). Moreover, requirements engineering principles and techniques are not only applicable to the design of technical systems (e.g. a software system), but can also help us to understand and improve non-technical systems (e.g. an organisation).

For requirements engineering to succeed in this exercise, we must be able to make informed decisions among alternative requirements and system designs. However, during RE no concrete materializations of an envisioned system (and its potential alternatives) are available, which limits our ability to assess their environmental performance and therefore to make informed decisions. We argue that it is nevertheless possible to assess the environmental performance of an envisioned system (even *early* in the requirements engineering process), by making use of *scenarios* and scenario-based requirements engineering techniques. In

particular, we describe two complementary techniques for assessing and comparing the environmental performance of alternative scenarios and how this can promote environmentally friendly decision making. This is aligned with existing work on the use of scenarios in the context of identifying and analysing non-functional requirements (e.g. in [2,3,4]).

In the following this paper (1) motivates scenarios in the context of environmentally aware system design, (2) proposes techniques for determining the environmental performance of scenarios, and (3) outlines how the former can form the basis for environmentally informed design decision.

2 Scenarios - snapshots of a environmental performance

A scenario is a storyline or script describing a system's behaviour in a particular situation of events. A scenario therefore contains information about the actions of an existing or envisioned system, in a particular context. The representation of a scenario can vary from a narrative description (a storyline) to a precise formal representation. For example, the scenario below is a narrative snapshot, in the context of a delivery company, told from the system perspective[1].

Scenario 1: *A parcel for Jim has arrived at Pit Street hub. The parcel is transported to Jim's home address. On arrival, Jim is not available and a notification message is left. The parcel is delivered to the closest pick up location, to be picked up by Jim.*

Scenarios are interesting in the context of environmentally aware system design, since they offer the right level of abstraction - their concrete representation of system behaviour (in the given example the system is the delivery company) *eases the correlation of environmental performance values.* Hence, scenarios allow us to not only get a behavioural snapshot of a system, but also a snapshot of its performance in a given situation. These snapshots are not sufficient to determine, e.g. the total carbon dioxide emission of a system for a particular period of time. However, we are not in the game of carbon accounting, but rather seek to support informed design decisions. When confronted with alternative scenarios, *it is sufficient to know which scenarios perform more preferred than others, to make environmentally aware decisions.*

Scenarios can not only be identified by observing the behaviour of a realized system, but also (1) *early* in the RE process, by envisioning the behaviour of a to-be system (e.g. see [5]) and/or (2) *later* in the RE process, by *extraction* from designs like an use case-, activity- or sequence diagrams (e.g. see [6]). In either case, for scenarios to form the basis for environmentally aware design decisions, their environmental performance must be explicated.

In short, to identify the environmental performance of a scenario, we first identify all (system) actions within the scenario. For example, the narrative scenario given above can be translated into a sequence of actions as shown in Figure 1. We then associate (by manual- or automated annotation) with each

---

[1] Scenarios can also be captured from the user perspective.

action a performance value, using one of the methods described in the following subsection. The overall performance of the scenario is then determined by accumulating all performance values along the sequence of actions.
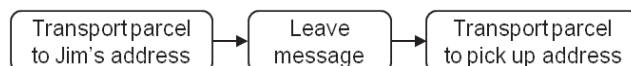
Fig. 1. The parcel delivery scenario as a sequence of actions

2.1 Identifying a scenario's environmental performance

In the following we describe two complementary techniques for correlating environmental performance values with actions of a scenario. This requires us to make precise the abstract notion of environmental performance. There are numerous ways in which "environmental performance" can be captured, i.e. *carbon dioxide equivalent* (CO2-e) *emission*[2], *water consumption, waste generation, damage to fauna and flora, air quality,* or some combination of the former. For ease of elaboration and without loss of generality, we use CO2-e as the only non-functional requirement of interest.

Educated guess: In this method the requirement engineer makes an educated guess on the expected CO2-e emission of each action of a scenario. Note that by guessing the CO2-e emission performance, the context of an action is taken implicitly into account. However, the quantitative amount of CO2-e emission (e.g. in number of kilograms) is hard to guess and in practice often leading to unrealistic values. We therefore recommend to abstract away from a quantitative scale to a qualitative scale. For example, the traffic light scale could be used, where red could denote a high CO2 emission impact, "orange" a moderate emission impact and green a low emission impact. We belief (and our observations confirm this) that practitioners have a good "gut-feeling" in guessing the CO2-e emission performance, when working with a simple scale. In the (likely) case that the assessment is done by more than one person, we further recommend to jointly do the initial assessments, such that a shared understanding of "high" and "low" emitting actions can emerge. A possible assessment of our running example (using the traffic light scale) is given in Figure 2.
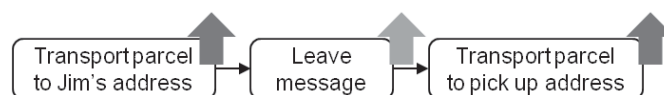
Fig. 2. Scenario assessment using the traffic light scale

---

[2] CO2-e is an expression of other greenhouse gases as their carbon dioxide equivalent by their global warming potential (CO2 itself has a global warming potential of 1).

This method is interesting in the case that (1) the envisioned system and context is still vague and as a consequence more detailed values cannot be determined, i.e. early in the requirements engineering process and (2) an initial "quick and dirty" overview of the performance of the scenario landscape is desired.

Modelling the resource context: More precise CO_2-e emission values can be determined, by considering the context in which an action is (or will be) performed. We argue that the relevant context for the environmental performance of an action is given by the resources it uses. More precisely, the emission values of an action are influenced by: (1) *What resources are used*, e.g. driving a truck with a particle filter causes less emission than driving the same truck without the particle filter; (2) *How the resource is used*, e.g. driving an empty truck causes less emission than driving a fully loaded truck; (3) *The intensity with which a resource is used*, e.g. driving a truck 100km or 200km; and (4) *What other sub-resources are used* e.g. the fuel used for combustion and the associated carbon emission for gathering and transporting the fuel to the petrol station (if this level of detail is desired - again we are not in the game of carbon-accounting).

In [7] a way of modelling this "usage-cost" interplay among resources (as well as other relationships like "is-a" and "part-whole" for other reasoning purposes) and actions is described. Essentially, the proposed resource model can be queried by a functional call, which states what resource is used, how it is used, and with which intensity, returning the respective performance values. For example, the call *use(truck, loaded, 30km)* (given a particular resource model instance) could return a value of 8.4kg CO_2-e emission. Given the former, each action in a scenario is annotated with a functional call. The expression is evaluated w.r.t. to the currently selected resource model instance (other instances could be considered to reflect an alternative context) and returns the corresponding emission figures. Figure 3 shows the running example with the annotation of functional calls. Note that values can also be annotated manually, e.g. the emission of the action "leave message" has been considered as neglectable and is therefore annotated with "0 kg CO_2-e".

Fig. 3. Scenario assessment using a functional call to a resource model

This method is interesting in the case that a decision among alternative scenarios is to be based on concrete and arbitrarily precise[3] CO_2-e emission performance values. Since the resources and their usage-cost relations need to be captured this method is more suitable later in the requirements engineering

---

[3] The more fine-grained the resource model the more precise its answers, but also the higher the cost for building and maintaining the model.

process.

Combining performance values: The CO2-e emission performance values associated with each action can now be used to determine the performance of a scenario. In case of quantitative CO2-e emission values, two values are combined by summation, such that the performance of a scenario is simply the sum over all values. For example, the quantitative CO2-e emission performance of scenario one is 9.8kg. In case of qualitative CO2-e emission values, two values are combined by selecting the least preferred, such that the performance of a scenario is simply the performance of its least performing action. For example, the qualitative CO2-e emission performance of scenario one is "high". Although, the later would treat two scenarios with values "high-high-high" and "low-low-high" as equally preferable, it allows us to treat both qualitative and quantitative measures in the same (algebraic) framework, i.e. the c-semi-ring framework [8]. This is important in the cases where some scenarios are given qualitative and others quantitative values.

2.2 Scenarios and environmentally informed decision making

An (environmentally aware) decision can be made, whenever there is choice - i.e. whenever it can be chosen among alternatives. In this paper we promote the use of scenarios as the basis of choice among alternative systems. Two different scenarios can be treated as alternatives, if they realize the same high-level stakeholder objectives (in which case the stakeholder objectives are treated axiomatically), *and/or* if they describe the behaviour of a system w.r.t. the same sequence of events. In the running example (which does not consider stakeholder objectives) the sequence of events is "parcel for Jim has arrived at Pit Street hub" before "Jim is not available". An alternative to scenario one, taking into account the same sequence of events, is scenario two (Figure 4 is a graphical description of the alternative scenario with associated qualitative and quantitative CO2-e performance values):

Scenario 2: `A parcel for Jim has arrived at Pit Street hub`. *Send mobile text message to Jim to confirm his availability on the expected arrival. Jim replies that he is* `not available` *during this time. The parcel is delivered to the closest pick up location, to be picked up by Jim.*

Applying the associated qualitative values, scenario one and two are equally preferred. However, applying the quantitative values, scenario two (total CO2-e emission of 7.65kg) is preferred over scenario one (total CO2-e emission of 9.8kg). Such preference relation among alternative scenarios can support environmentally aware decision making and system design at least in the following. (1) The chosen set of scenarios can be used to extract new requirements. A way of deriving requirements from scenarios has, for example been described in [9]. (2) The chosen set of scenarios can be used to analyse existing requirements against the set of preferred scenarios (e.g. see [10]), which can then form the basis for adapting the existing requirements. However, in all cases the decision for a particular set of requirements must take into consideration the impact on other functional
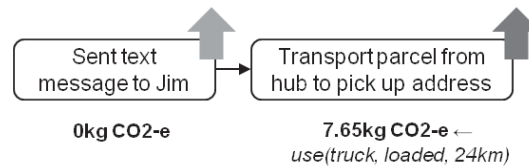
Fig. 4. Alternative scenario with concrete and abstract CO2-e performance values

and non-functional requirements, i.e. the global impact of a particular decision must be understood.

3   Conclusion & Future Work

This paper motivates the use of scenarios as a basis for building environmentally sustainable systems. In this context, two complementary techniques, which can be used to assess the environmental impact of scenarios have been described as well as how this can form the basis for environmentally aware decision making.

Future work is concerned with the following question. Given a set of (environmentally preferred) scenarios describing a to-be system, how can an existing system design be *minimally changed*, such that it is shown to entail all to-be scenarios. Minimal change is important, because it protects existing investments in the context of desired change. We seek to answer this question by leveraging "light-weight" formal machinery (limiting the burden on the engineer).

References

1. Stefan, D., Letier, E., Barrett, M., Stella-Sawicki, M.: Goal-oriented system modelling for managing environmental sustainability. In: 3rd Workshop on Software Research and Climate Change. (2011)
2. Sutcliffe, A., Minocha, S.: Scenario-based analysis of non-functional requirements. In: REFSQ. (1998) 219–234
3. Gregoriades, A., Sutcliffe, A.: Scenario-based assessment of nonfunctional requirements. IEEE TSE **31**(5) (2005) 392 – 409
4. Nixon, B.: Management of performance requirements for information systems. Software Engineering, IEEE Transactions on **26**(12) (2000) 1122–1146
5. Hooper, J., Hsia, P.: Scenario-based prototyping for requirements identification. In: ACM SIGSOFT Software Engineering Notes. Volume 7., ACM (1982) 88–93
6. Briand, L., Labiche, Y.: A uml-based approach to system testing. Software and Systems Modeling **1** (2002) 10–42
7. Hoesch-Klohe, K., Ghose, A.: Towards Green Business Process Management. In: SCC. (2010)
8. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based constraint satisfaction and optimization. Journal of the ACM (JACM) **44**(2) (1997) 236
9. Alrajeh, D., Ray, O., Russo, A., Uchitel, S.: Extracting requirements from scenarios with ILP. Lecture Notes in Computer Science **4455** (2007) 64
10. Sutcliffe, A.: Scenario-based requirements analysis. RE (1998)

Green Requirements Engineering with the GREENSOFT Model
Taking the whole Lifecycle of Software into Account

Eva Kern, Markus Dick, Stefan Naumann, Timo Johann, Matthias Giesselmann, Patrick Lang

Umwelt-Campus Birkenfeld, Trier University of Applied Sciences, Institute for Software Systems
greensoft@umwelt-campus.de

1   Green and Sustainable Software Engineering

In an earlier paper we gave the following definition: "Sustainable Software Engineering is the art of defining and developing software products in a way so that the negative and positive impacts on sustainability that result and/or are expected to result from the software product over its whole lifecycle are continuously assessed, documented, and optimized."

Based on that definition it is required to pay attention to the whole life cycle of a software product from beginning on, starting with the requirements review. Since many different processes, products and services are involved in this life cycle, which have impacts on sustainable development, they must be considered in order to figure out if a software product and even its engineering process is green or not. In view of the fact that several design and implementation decisions are made in the requirements phase, it is necessary that the consequences of these decisions are taken into account at this phase.

2   Reference Model for „Green Software"

Based on this aspects we developed a conceptual reference model shown in our multi-media presentation that supports sustainable production and usage of software. It includes a life cycle of software products, sustainability criteria and metrics for software products, procedure models as well as recommendations for actions and tools for purchasers, developers, administrators, and users. In that way the different user roles are addressed.

The introduced *Lifecycle for Software Products* supports responsible persons in estimating the impacts on sustainable development by software products. The approach based on Life Cycle Assessment (LCA) [1] takes the direct effects (Green IT) and the indirect effects (Green by IT) into account.

The quality model (based on [2–4]) gives an overview of potential aspects which can be taken as *Sustainability Criteria and Metrics for Software Products*. The metrics need to be defined for specific types of software. In order to support software developers during the development process and administrators and users in

configuring or choosing software we present a measurement model. The method is to compare the energy consumption of different software or different configurations of software.

The generic *Procedure Model* takes an organizational perspective look at the development phase of a software product and extends software development processes by sustainability aspects.

As examples for *Recommendations for Actions and Tools* the model includes a knowledge base with a collection of guidelines, tips and hints in the area of sustainable information technology. Regarding the Green Web the Firefox Add-on "Green Power Indicator" displays whether the called site is hosted on a server, which is operated with environment-friendly produced electricity.

3   Conclusion

We present a conceptual reference model for Green and Sustainable Software that comprises a software products' life cycle, direct and indirect effects, different user roles and approaches for activities. As a reference model its objective is to structure concepts, strategies, activities, and processes of Green Software Engineering and to organize research in the field of Sustainability Informatics. With our model, requirements engineers can take different aspects of sustainable and green software into account. This comprises e.g. aspects like software architecture decisions, tools for measuring energy-efficiency code and what impact each software engineering phase onto environment has.

4   References

1. Deutsches Institut für Normung e.V. (2009) Environmental management - Life cycle assessment - Principles and framework (ISO 14040:2006); German and English version EN ISO 14040:2006. Beuth, Berlin 13.020.10(DIN EN ISO 14040:2009-11 (D))
2. Albertao F, Xiao J, Tian C, Lu Y, Zhang KQ, Liu C (2010) Measuring the Sustainability Performance of Software Projects. In: 2010 IEEE 7th International Conference on e-Business Engineering (ICEBE 2010), Shanghai, China, pp 369–373
3. Naumann S, Dick M, Kern E, Johann T (2011) The GREENSOFT Model: A Reference Model for Green and Sustainable Software and its Engineering. SUSCOM 1(4):294–304. doi:10.1016/j.suscom.2011.06.004
4. Taina J (2011) Good, Bad, and Beautiful Software - In Search of Green Software Quality Factors. CEPIS UPGRADE XII(4):22–27

Integrating the Complexity of Sustainability in Requirements Engineering

Martin Mahaux[1], Caroline Canon[2]

[1] PReCISE Research Centre, University of Namur, Belgium
[2]Sustainable Development Research Group, University of Namur, Belgium
`{martin.mahaux, caroline.canon}@fundp.ac.be`

Abstract. **[Context and Motivation]** While having a simple definition, Sustainable Development is a broad, interdisciplinary and complex concept. Applying this concept when designing products is therefore a complex task that requires a lot of interdisciplinarity. **[Question/Problem]** As software continues to invade all aspects of our lives under ever-renewed forms, we realize that designing sustainable software is probably of paramount difficulty and importance. **[Position]** This position paper argues that this new field will have no other option than integrating this complexity into its design practices through opening collaborations with sustainability experts.

2. Introduction

Sustainability Informatics has been suggested as a new research field in 2010 [1]. It is born out of the Environmental Informatics field, which is now comprised within Sustainable Informatics. Within this discipline, Sustainable Software has received a significant attention. Results have been mainly published in specialized venues, of which a nice summary can be found in [2]. In this publication, Naumann et al. combine many existing works, as well as environmental sciences knowledge, to lay solid foundations for studying Sustainable Software. Their holistic study result in new definitions for Sustainable Software and its Engineering, as well as in a framework for designing sustainable software called the GreenSoft Model. It specifies where to look for software impacts on sustainability and makes initial suggestions on how to measure them and how to deal with them according to your process and role regarding software. This is, to our knowledge, the most advanced and comprehensive model of the genre to date.

However, while certainly containing useful material, we still consider it as a mostly empty box, that will have to be filled with more concrete techniques and tools for designing sustainable software. In particular, we noted that the question of the complexity of the sustainability concept and how to integrate this complexity into already complex software engineering is mentioned, but escaped, rather silently.

3. Sustainability: a complex concept.

The university of Namur (FUNDP) has recently set up an interdisciplinary research group around sustainability. It is pursuing mainly 4 research directions, one of them being centered on the definition of the sustainability concept. When the Computer Sciences oriented authors of this paper invited this group to collaborate, they expected to receive answers. Instead they realized there were no simple answers, and that complex answers were not ready yet.

The Sustainability Research Group is composed of researchers in Human and Nature Sciences, aiming at elaborating a map of research in "Sustainable development". What is in fact a research in Sustainability? What are the criteria to say that a research concerns Sustainability? Realizing that each discipline had a specific viewpoint on sustainability, they decided to start with having each discipline to present his viewpoint and discuss it. Divergences and convergences are carefully kept aside for later reconciliation. The first and only current result is that researchers are now aware that a long time will be needed in order to answer these questions, due to the intrinsically interdisciplinary nature of the sustainability concept. Our position is that Requirements Engineers should follow on these results and collaborate in order to translate them to their own discipline.

Notwithstanding this, research has already delivered frameworks to analyze sustainability. The famous Life Cycle Analysis (LCA) framework, used in [2], is a prominent example, but its scope is quite limited. More complex models can also be found, see for example: [3–6]. They're all incomplete as any model is, but here particularly as they usually result from mono-disciplinary efforts. They however offer interesting tools to requirements engineers, and we stand behind the position that research in sustainable requirements should take the time to investigate these and translate them to it's body of knowledge, similarly to what Naumann et al. have started to do with LCA and the GreenSoft Model.

4. Requirements Engineering and impacts on the software life-cycle.

4.1. The GreenSoft Model

The first part of the GreenSoft model [2] recalls that software impacts sustainability all along its lifecycle (Development, Usage, Disposal), at least at three levels:
First-order impacts are direct effects [like…] resource use and pollution from mining, hardware production, power consumption, and disposal of electronic equipment waste. Second-order impacts are effects that result indirectly from using ICT, like energy and resource conservation by process optimization (dematerialization effects), or resource conservation by substitution of material products with their immaterial counterparts (substitution effects). Third-order

impacts are long term indirect effects that result from ICT usage, like changing life styles that promote faster economic growth and, at worst, outweigh the formerly achieved savings (rebound effects)[2].
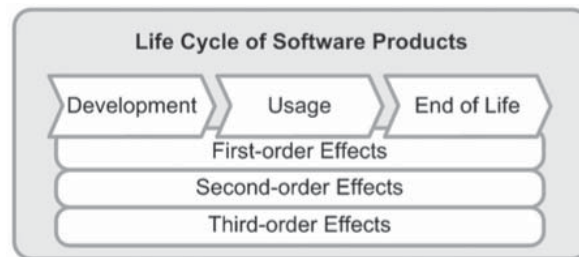
Figure 1: Software Life Cycle and impacts on sustainability [2]

The paper also insists on the fact that second- and third-order effects might well be the most important, but the harder to grasp. The distinction between software that has a sustainability-related main purpose and other-purpose software is also highlighted. It is argued that second- and third-order effects are nearly impossible to grasp in the latter case.

In this section we use the first part of the GreenSoft Model to briefly see where Requirements Engineers should take care about sustainability impacts. First we discuss the phase (development, usage, disposal), then the level of impact (1st, 2nd or 3rd order).

4.2. The Requirements Engineer's Point of View

RE is obviously primarily concerned by the usage phase of the software. But RE can also reduce the relative impact of the development and disposal phase: by enabling software to last longer. This in turn relates to qualities such as reliability, adaptability, maintainability or context-awareness of software. While specific development paradigms such as Agile claim their share of the pie in this area [7], it is clear that the fitness for purpose of the software is the prime quality that will save it from being thrown in the bin too early. Consequently, a correct requirements engineering work has a lot to do with software that lasts.

So far as software is concerned, fighting negative 1st-order impacts means designing "lean" software: software that will consume just what it needs in terms of energy and hardware. While programming languages and techniques have a predominant impact here, the requirements work also plays an important role. Keeping the software to functionalities that are strictly needed is key. Variability management techniques can also help software engineers to offer more customizable products, so users can select what they need and only this, removing unused features and associated energy costs.

Caring about 2nd- and 3rd-order effects means designing software that induce more sustainable human behaviours. For any software, the functionalities that we design may have an impact on sustainability. The Requirements Engineer is the most appropriate person to integrate sustainability at this time. But this won't be easy, as the complexity of software is multiplied by the complexity of sustainability and human behaviour. For example, e-bay, which fosters reuse of physical goods (positive impact), may very well foster over-consumption (negative rebound). It's functionality to show goods that are close to your home saves on transport impacts, but the one that shows you results from far away has the reverse effect. E-bay fosters individual exchanges between people, and provides a sense of community, bringing people together, which seems to be positive. But is it really so? Social networking tools in general, a prominent example, have a clear impact on social sustainability of our society. But how can we measure this impact? How can we assess if it serves a more or less sustainable society?

In an experience report, Mahaux et al. [8] show that Requirements Engineers can take the time to assess at least second-order effects of a business-oriented software. They experimented with very concrete adapted techniques and highlighted how Requirements Engineers needed to talk to Sustainability specialists in order to master the complexity of this domain and integrate it into their developments. Just as Requirements Engineers do with other quality requirements like security [9], they have to tailor specific techniques and craft the collaboration between Requirements Engineers and other disciplines specialists to reach the desired quality levels. In [10], Cabot et al. propose to consider sustainability as a high level goal amongst others, and using goal-oriented techniques to help decision-making for Requirements Engineers and stakeholders. They also observe that the first problem is the lack of standard definitions for sustainability concepts, and suggest Requirements Engineers should work on defining taxonomies for this concept.
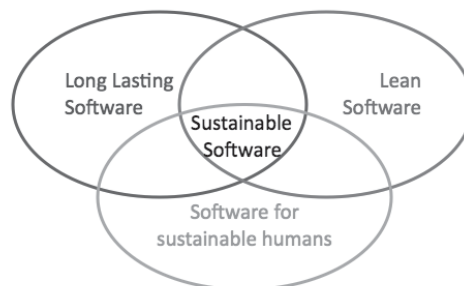
Figure 2: Areas for action for Requirements Engineers

5. Conclusion

Requirements Engineers have a role to play in order to make software more sustainable. It encompasses efforts to build lean and long lasting software, but also software that helps systems using it to be more sustainable. To do so they first need to connect with research that will let them understand what is a sustainable society. Indeed, the complexity of this topic should not be underestimated and, while some simplifying frameworks are useful and needed, integrating the real complexity of the sustainability concept will require more work. Researchers from both disciplines should work collaboratively to develop adequate frameworks for understanding sustainability in RE and efficient tools to take decisions for building sustainable software. How these interactions might work, which sustainability experts should be integrated, which role plays the client who orders the software, in which part of the RE process is this collaboration in particular useful… are good examples of the coming research questions in this direction.

6. References

[1] Naumann, Stefan: Sustainability Informatics: A new Subfield of Applied Informatics? In: Mûller, Andreas; Page, Bernd; Schreiber, Martin (Eds.): EnviroInfo 2008. Environmental Informatics and Industrial Ecology, 22nd International Conference on Environmental Informatics. Aachen 2008

[2] 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*, vol. 1, no. 4, pp. 294-304, Dec. 2011.

[3] P. Ekins, S. Simon, L. Deutsch, C. Folke, and R. De Groot, "A framework for the practical application of the concepts of critical natural capital and strong sustainability," *Ecological Economics*, vol. 44, no. 2-3, pp. 165–185, 2003.

[4] S. López-Ridaura, O. Masera, and M. Astier, "Evaluating the sustainability of complex socio-environmental systems. The MESMIS framework," *Ecological indicators*, vol. 2, no. 1-2, pp. 135–148, 2002.

[5] E. Ostrom, "A general framework for analyzing sustainability of social-ecological systems," *Science*, vol. 325, no. 5939, p. 419, 2009.

[6] "Reliable Prosperity - A pattern language for sustainability." [Online]. Available: http://www.reliableprosperity.net/. [Accessed: 26-Jan-2012].

[7] K. Tate, *Sustainable Software Development: An Agile Perspective*, 1st ed. Addison-Wesley Professional, 2005.

[8] M. Mahaux, P. Heymans, and G. Saval, "Discovering Sustainability Requirements: An Experience Report," in procs *REFSQ*'11, pp. 19–33.

[9] D. Firesmith, "Engineering Safety and Security Related Requirements for Software Intensive Systems," in *ICSE Companion*, 2007, p. 169.

[10] J. Cabot, S. Easterbrook, J. Horkoff, L. Lessard, S. Liaskos, and J. N. Mazón, "Integrating Sustainability in Decision-Making Processes: A Modelling Strategy," in *31st International Conference on Software Engineering-Companion Volume, 2009. ICSE-Companion 2009*, 2009, pp. 207–210.

RE4ES: Support Environmental Sustainability by Requirements Engineering

Birgit Penzenstadler[1], Bill Tomlinson[2] and Debra Richardson[2]

[1] Technische Universität München, Germany
`penzenst@in.tum.de`
[2] University of California, Irvine, US
`wmt@uci.edu, djr@ics.uci.edu`

Abstract. [**Motivation:**] Environmental sustainability is an important concern. Information and communication technology (ICT) innovation is ambivalently positioned with regard to our rapid development and shortening innovation cycles. On one hand, information technology facilitates the (excessive) usage of resources. On the other hand, ICT can also help to significantly reduce human impact on the environment.
[**Problem:**] Environmental sustainability is currently not supported explicitly in requirements engineering (RE). This leads to the problem that (a) environmental sustainability is not yet given sufficient importance and (b) it is difficult to manifest in requirements & design and therefore hard to assess.
[**Principal idea:**] We need to combine the knowledge of RE, environmental informatics, and further disciplines, to develop an RE approach that tailors analysis, documentation, and assessment for ICT systems where environmental sustainability is a first class quality objective.
[**Contribution:**] This paper is a research preview on an approach to help requirements engineers handle sustainability as a first class quality objective. It elaborates on how we plan to refine and validate this approach in the future.

Keywords: requirements, sustainability, environment, requirements engineering, quality modeling

1 Introduction & Motivation

The most cited definition of sustainability is to "meet the needs of the present without compromising the ability of future generations to meet their own needs" [1]. Although our approach primarily aims at environmental sustainability, it must also be socially (and economically) sustainable in order to have practical significance [2]. As Mahaux [3] pointed out, we need a toolbox for supporting it in requirements engineering. We extend the idea of such a toolbox in this research preview and provide some of our drafts.

Problem: The use of information and communications technology (ICT) contributes significantly to the usage of our planet's resources [4]. However, ICT

bears a lot of potential for "greening through IT" [5] by making our life more environmentally sustainable by technological support for our daily life; this is the context of our research. In contrast, Green IT or "greening of IT" is making hardware and software of ICT systems more resource-efficient; we do not focus on this. We must improve the environmental sustainability of humankind to protect our living space for future generations. Missing is a comprehensive understanding of how software engineering, and especially requirements engineering (RE), can help in this endeavor.

Contribution: We are analyzing what and how RE can contribute to the improvement of the environmental sustainability of ICT. We primarily focus on the development of ICT systems that have environmental sustainability in their explicit system vision (and abbreviate these systems with ICT4ES), because we assume the stakeholders of such systems to be more willing to adapt their development processes according to that quality objective. Our goal is to support the ICT4ES development with an adequate requirements engineering approach that integrates the knowledge of environmental informatics. This enables software engineers to handle sustainability as first class quality objective. Our research questions are:

RQ1: What are the implications for RE of ICT4ES, i.e., when making environmental sustainability a first-class quality objective for development?
For ICT4ES as we defined the term, environmental sustainability is an overall development goal. However, it is not clear how that impacts the requirements for a system. We seek to understand what is necessary to be taken care of when developing ICT4ES and how the business processes and business goals differ from those of traditional products.

RQ2: How can the necessities resulting from ICT4ES be implemented in an RE approach?
We aim at a toolbox to support the demands resulting from the goal of contributing to environmental sustainability. First, we analyze which artifacts are necessary to document the newly arising demands and what their concrete contents are. Then, we investigate which concepts have to be supported and which methods are required to elaborate these artifacts and how they have to be adapted.

RQ3: How can we assess the impacts of a given software system for environmental sustainability, including both direct and indirect effects, and considering different groups of stakeholders?
We elaborate metrics to measure environmental sustainability and provide an answer as to how a system can be proven to fulfill the sustainability requirements imposed upon it. Furthermore, we investigate an appropriate way to translate the requirements into acceptance criteria and how these criteria can be incorporated into an overall quality model.

2 Related Work

Sustainability is beginning to play an important role in software engineering, with the RE'08 keynote, the ICSE'09 Software Engineering for the Planet spe-

cial session, the CAiSE'10 panel, the WSRCC 2009, 2010, and 2011, and the conference slogan for ICSE'12. The first author of this paper completed a systematic literature review on sustainability in software engineering [6].

Amsel et al. [7] discuss ideas on how to support sustainability in SE. Cabot et al. [8] performed a case study for sustainability as goal for the ICSE organization with i* models to support decision making for future conference chairs. Naumann et al. [9] investigate how web pages can be developed with little environmental impact, i.e., energy-efficiently, and work on a respective guideline for web developers. Mahaux et al. [3] performed a case study on a business information system for an event management agencyto assess how well some current RE techniques support modeling of specific sustainability requirements.

These works look at either a specific application domain or a specific development technique and adapt them to support sustainability modeling, while this project aims at an encompassing approach to be evaluated in various domains of ICT4ES systems. No other work yet proposes solutions for how to support quality modeling of environmental sustainability for software systems.

3   Approach to RE for ICT4ES

Our approach to RE for ICT4ES is planned in two phases: First, we conduct an analysis of domains as well as values and goals of the respective stakeholders, then we design a tailored RE method that supports the gathered specifics for ICT4ES (see Fig. 1). All activities described in this section are in progress, which means we have started but not yet completed them.

3.1   Analysis of Domains, Values, and Goals

Environmental sustainability can be supported by software systems in different ways, e.g., (a) information systems for environmental sciences, including climate models, earthquake warning, etc., (b) information systems that support green business processes, for example environment-friendly event management, and (c) embedded systems that lower our energy consumption. Therefore, we need to analyze the different types of domains that need support in explicitly addressing environmental sustainability in their software engineering approaches.

Based on the distinction of domains, we perform structured interviews in industry and academia with representatives from different domains. The interviews are followed by a systematic analysis and an interpretation that draws conclusions for the design of the envisioned method's elements.

Starting with the results of the interview analysis, we elaborate a map of values for environmental sustainability and we detail the goals in a taxonomy, focusing on the ones that relate to requirements engineering for ICT4ES systems:

Value map for environmental sustainability in SE (RQ1) The value map shall put the value of sustainability into relation with traditional software engineering values as in the framework described by Khurum [10]. Her framework

relies on data gathered in interviews with practitioners and allows to create impact evaluation patterns from value maps.

Goal taxonomy for sustainability in SE (RQ1) The goal taxonomy decomposes and details the aspects of environmental sustainability from the point of view of software engineering. The input is the value map and for each value we can deduce supporting goals. Initially, most of these goals are independent of the system to be developed. Each of the goals is then decomposed hierarchically until the goals are sufficiently specific to be transformed into requirements.
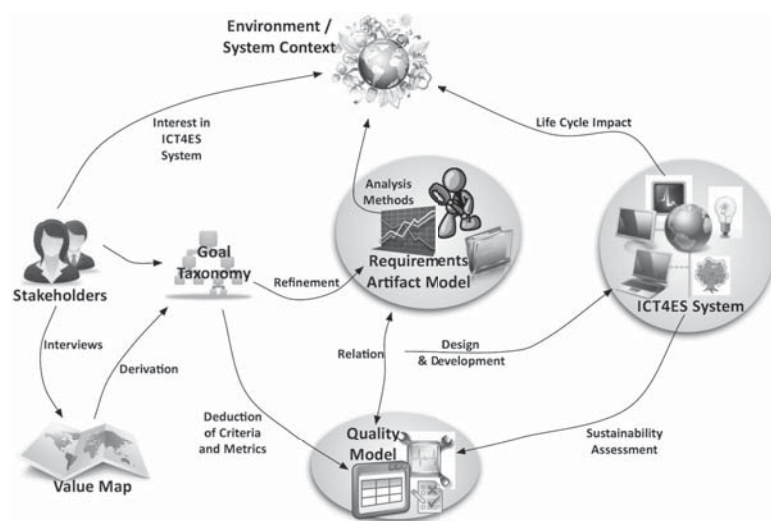
Fig. 1. Environmental Sustainability in Requirements Engineering.

3.2 Design of a Tailored RE Approach

From the goal taxonomy, we gather requirements for artifacts, methods, and models for the documentation of sustainability requirements arising by deduction from the goal taxonomy with respect to a specific ICT4ES system. Based on these requirements and the knowledge acquired in the earlier phases of the project, we conduct an analysis and evaluation of different techniques, compare existing approaches, and develop a tailored RE approach including a quality model that provides indicators and metrics to assess environmental sustainability.

Sustainability requirements artifact model (RQ2) An artifact model gives guidance on structure and content to be elaborated when documenting sustainability requirements and related information like environmental impact, stakeholders, rationale, etc. Based on our experience [11], we develop an artifact model for representing sustainability requirements and related information.

Adapted analysis techniques (RQ2) To transition from goals to requirements and to adequately document these requirements according to an artifact model, we elaborate analysis techniques and documentation methods that form part of an RE approach tailored to ICT4ES. Solutions include adaptations of creativity techniques, life cycle analysis, environmental impact assessment and risk analysis techniques as well as handling of environmental information in form of data, statistics, and models.
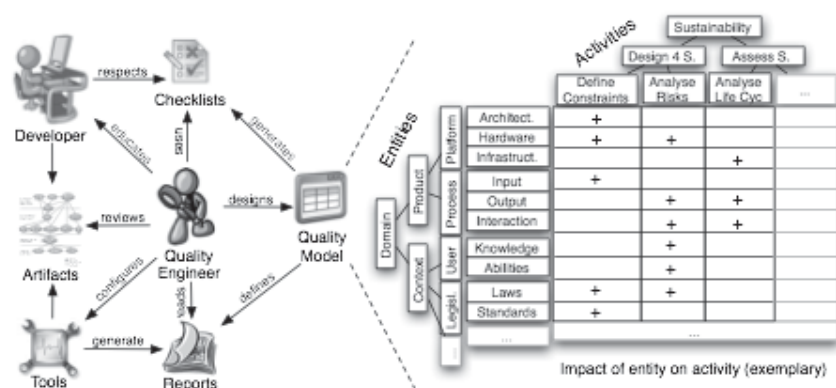
Fig. 2. Model-based Quality Assurance (adapted from [12]) & Quality Model Excerpt.

Deduced quality model (RQ3) The quality model is built upon the input from the value map and the goal taxonomy. A quality model is a model with the objective to describe, assess and/or predict quality [12]. The activity-based quality model is elaborated on the basis of concepts proposed in [13]. It includes criteria for sustainability assessment as well as indicators and metrics to evaluate and measure a software system's compliance to the sustainability requirements. Fig. 2 shows the model-based principle and an excerpt of the quality model draft.

Case studies (RQ1-3) The approach will be evaluated in industrial case studies, including the value map, the goal taxonomy, the artifact model, the analysis techniques, and the quality model. The qualitative evaluation will be implemented as a comparative study. The case study already under way is on car sharing; another one will be on an irrigation system.

4   Conclusion

In this research preview, we have introduced our ongoing research on a tailored RE method for ICT systems for environmental sustainability. The analysis phase investigates the domains and elaborates values and goals with the respective stakeholders. The design phase provides a tailored artifact model with analysis

methods and a deduced quality model. Both will be evaluated in industrial case studies. We are preparing a guideline for the industry interviews and evaluate approaches from related disciplines in student seminars as described in [14] for preliminary studies.

Our contribution will provide software engineers with a toolbox to handle sustainability as first class quality objective. This enables "greening through IT" — to produce ICT systems that have positive impact on their surrounding eco-systems and therefore not only meet the needs of the present (by satisfying traditional quality objectives) but at the same time preserve the ability of future generations to meet their own needs (by meeting sustainability quality objectives). As software systems have a profound influence on many different facets of global civilization, including sustainability in the design of these systems has the potential to have transformative impacts on the world in which we live.

Acknowledgments: We would like to thank Martin Mahaux for providing feedback on an earlier version of this paper.

References

1. Brundtland et al.: Our Common Future. In: UN Conference on Environment and Development. (1987)
2. Sverdrup, H., Svensson, M.G.E.: Defining the concept of sustainability. In: Systems Approaches and Their Application. Springer (2005) 143–164
3. Mahaux, M., Heymans, P., Saval, G.: Discovering Sustainability Requirements: an Experience Report. In: 17th REFSQ. (2011)
4. The Climate Group: Smart 2020: Enabling the low carbon economy in the information age. Technical report, Global eSustainability Initiative (2008)
5. Tomlinson, B.: Greening through IT. MIT Press Association (2010)
6. Penzenstadler, B., Bauer, V., Calero, C., Franch, X.: Sustainability in Software Engineering: A Systematic Literature Review. In: 16th Intl. Conf. on Evaluation and Assessment in Software Engineering. (2012)
7. Amsel, N., Ibrahim, Z., Malik, A., Tomlinson, B.: Toward sustainable software engineering. In: Proc. of the 33rd Intl. Conf. on Software Engineering. (2011)
8. Cabot et al.: Integrating sustainability in decision-making processes: A modelling strategy. In: 31st Intl. Conf. on Software Engineering. (2009) 207 –210
9. Naumann, S., Dick, M., Kern, E., Johann, T.: The greensoft model: A reference model for green and sustainable software and its engineering. Sustainable Computing: Informatics and Systems (2011) –
10. Khurum, M., Gorschek, T.: Software value map - an exhaustive collection of value aspects for the development of software intensive products (2011)
11. Fernandez, D.M., Lochmann, K., Penzenstadler, B., Wagner, S.: A case study on the application of an artefact-based requirements engineering approach. In: 15th Intl. Conf. on Evaluation and Assessment in Software Engineering. (2011)
12. Wagner, S., Deissenboeck, F., Winter, S.: Managing quality requirements using activity-based quality models. In: Intl. Workshop on Software Quality. (2008)
13. Winter, S., Wagner, S., Deissenboeck, F.: A comprehensive model of usability. In: Proc. of Engineering Interactive Systems. (2007)
14. Penzenstadler, B., Fleischmann, A.: Teach sustainability in software engineering? In: 24th Intl. Conference on Software Engineering Education & Training. (2011)

Writing Requirements for Electromobility and Smart Grids Systems: Challenges and Opportunities

Jean-Charles Jacquemin[1], Martin Mahaux[2]

[1] CERPE, University of Namur, Belgium,
[2] PReCISE, University of Namur, Belgium,
{Martin.Mahaux, Jean-Charles.Jacquemin}@fundp.ac.be

Abstract. If they are to deliver their promises without creating the need to replace the investments we made in the electric grids in the last decades, electric vehicles, electric grids and their users will have to work together in a smart way. We present some opportunities and challenges that lie behind this for requirements engineers, and stand behind the position that this matter should be part of their research agenda related to sustainability.

1. Introduction

The renewed interest in electromobility was considered some years ago as a simple paradigm shift in the automotive sector. In this vision, an Internal Combustion Engine (ICE) vehicle was simply transformed in an Electric Vehicle (EV) by removing the fossil fuel engine to replace it by an electric motor. After all, that was the situation in the early years of the XXth century. However the need to reduce both the imported oil dependency and the emissions from the transportation sector changed this view [1].

In the same time, and for similar reasons, power utilities are also experiencing an important shift. While they have built their reputation on the reliability and security of supply through years of incremental innovations, as we move into the XXIst century it is evident that the distribution systems concepts are approaching their limits. The need to incorporate an ever-increasing amount of renewable sources - such as wind and solar - as well as distributed generation is changing the game. Today, electric distribution systems are still being designed in an hierarchical model similar to what was the practice in Computer Networks during the 70's, and it is widely recognized that they will have to evolve to a "Energy Web" model, bringing some of the attributes of the Internet to energy distribution. What is needed is more flexibility, implementing features like "plug-and-play" and "peer-to-peer" operation, which we have learned to take for granted in the Internet [2].

Distributed generation of renewable energy as well as electromobility appeared as two problems for the current electric grid. Integrating adequate ICT systems into it,

making it a "Smart Grid", has the potential to transform these two problems in a set of opportunities. This is the promise that Smart Grids will have to deliver, and this will demand smart requirements engineers.

2. Which new ICT systems ?

In this section we define briefly where new ICT systems will have to be integrated into the grid, and what is so smart about it.

2.1. Smart charging.

The Electric Vehicles (EVs) will represent a new kind of load for the electric network, with a stochastic behaviour in time and space. An overload of the power system (in its generation, transmission or distribution components) may occur due to the simultaneous charging of vehicles. Smart Grids may provide more clever solutions than just oversizing the system; they will enable "smart charging", supplying the power according to the availabilities of the power system. Consequently, any charging point will need information about these availabilities [3].

2.2. Storing renewable energies.

On the other side, the storage capacity represented by a float of EVs may, in the future, become a strong enabler of the introduction of large amounts of renewable energy into the system. Electric vehicles would be equipped with a plug for connecting to the Mains and another to connect to the Net. When the vehicle will be parked at night, at home, it will be connected with both plugs, and it will be connected again, in the morning, when parked at the office's garage. While parked, the vehicles will keep receiving information about the incremental costs of energy. They will store energy in batteries when it is cheap as there is a lot of wind and solar energy available, and will sell back the energy when the price is high enough, due to the scarcity of production. An energy reserve will be kept, in order to enable the users to continue using the vehicle for the day-to-day needs. Parked in the garage, electric vehicles will, in the future, help pay themselves by arbitrating on the price of energy. A simulation of this principle in Belgium can be found in [4]. Again, many intelligence and information is needed.

Battery swap stations are a particular case because the storage of renewable energies is centralized in the station which can better accommodates the volatility of renewable energy supplies [5]. Given the specific situation of the reserve of batteries in the station, it can also have a significant role as a buffer for load fluctuations in the network, while removing the EV user anxiety about the battery wear and tear. ICTs are needed to correctly manage both the energy flows and the EV driver's usage (both in terms of energy consumption as financially) of the station.

2.3. Peer-to-peer charging stations

A third domain of prime interest is the necessity for EV users to have access to a sufficient infrastructure of charge points. Public investment appears too costly, too slow and inefficient. Given this fact, new initiatives of charge infrastructure sharing appear as Plugshare [6] in the US, or Plugsurfing [7] in Europe. Both initiative use ICTs to provide information on smartphone applications or on the Internet about characteristics, status and location of private and public charging points and offer GPS guidance as well as payment management services.

2.4. Connectivity in the EV

The last domain, less specific in some aspects to EVs only, is the integration of advanced connectivity services in the e-mobility. It concerns bringing content into the car, enabling seamless communications to and from it, and controlling your home from your car. But also technologies helping the user to drive more safely and more ecologically, including auto collision avoidance, lane drift assistance, parking, speed monitoring, hands-free, text-to-voice, driver drowsiness detection, remote diagnosis by the vehicle manufacturer and more [8]. According to Deloitte's recent survey [9], those features will be highly demanded by the next generation of drivers.

2.5. Efficient Electricity Markets

To be efficient, markets must get reliable information at the right time. On the supply side of the market, they need information about the weather, to foresee renewable energy generation, as well as information about which energy is stored where. The detection of incorrect use of storage facilities, to avoid a possibly destabilizing speculation for the only profit of one actor, will require more information. On the other hand, patterns of EV drivers' behavior must be estimated to correctly predict the demand side of the market. Both market sides thus need constant flows of information to build correct anticipations of equilibrium situations and price levels. The vision of an important Electricity producer in Germany can be consulted in [10].

3. Writing Requirements for those new systems.

Redesigning the very complex electricity system will involve a huge requirements effort. There are many stakeholders involved, and many aspects of our societies are concerned. While it seems clear that most of the technological components are available today, writing effective requirements for these systems still look like an important challenge. Below we list a few of the challenging questions that live around this system, grouped by the class of stakeholder they belong to. The rich picture below gives an overview of these actors and their principal relations with the grid. It is freely inspired from [10], [11].
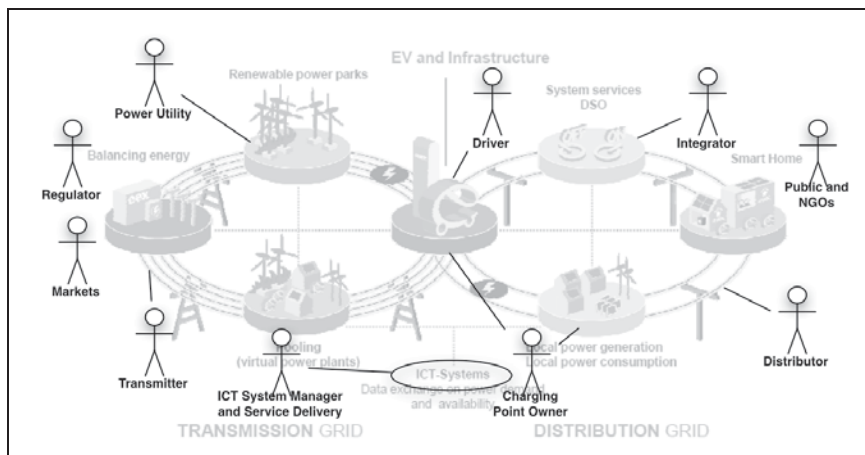
Figure 1: EV-centered smart grid and its main actors

Regulator: How to ensure consumer choices and legal rules are respected in the context of a liberalized electricity market, in particular the free choice of a given producer, the free choice of a specified pricing scheme? How to deal with rapidly evolving laws and regulations as we design our systems around it? How will we deal with technological monopolies (e.g. charging/swapping stations)? How will we enforce interoperability?

Driver: How will he manage his EV, minimizing its cost, maximizing its financial return, and still using it as a reliable vehicle? How to deal with uncertainties (potential mobility emergencies)? Will people allow to be deprived of their vehicle use if rewarded enough? Or if no other choice? How to change a pre-assigned (dis)charging scheme in case of uncertainties, in which timeframe? How to choose a provider? Where to charge? Is the driver ready to make the daily effort needed to manage this effectively? Or will he ask someone else to do this?

Power Utility: How to manage this new complexity and still ensure reliable and green power to people in this dynamic environment, for the lower cost? How will he be able to monitor the state of the system? Which available (un)conditional storage capacity may be used on the spot? How to foresee the demand in electricity? How to ensure revenues in this dynamic world?

Integrators: it is already clear that third party operators like integrators will take a great importance in providing services to users and perhaps producers and or distributors; the main question is: how to guarantee impartiality, integrity and confidentiality on the data and their use?

Markets: When the grid needs to buy energy, where will it take it? From who? At what price? When many users need energy, who will receive it first? At what price?

What is automatic and what not? How to organize and respect the equality of treatment of users? Are there various priority levels? Various Quality of Services? Morally, can we deprive a low priority user who has to drive to the hospital? How to ensure a proper operation of the market while keeping confidentiality on private data?

Transmitters: While they seem to be less impacted by the EV introduction if the downside of the market is well organized, some transmitters show interest in the storage capacities of battery swapping stations, as they intend to use those capacities to regulate the high and medium voltage power systems. For example, Elia, operating the Belgian transmission system, takes part in the eMobility project, "Greening European Transportation Infrastructure for Electric Vehicles" [12]. On the other hand, the exchange of information with low voltage distributors will represent vital statistics for a good operation of the whole system and stable electricity system.

Charging point Owner: How to share my CP? To who? Again, who's first? Accounting: how to manage electricity bills of both the EV user and the charge point owner? How to manage payments (included the potential problem of VAT). Liabilities: who is legally responsible of potential damages to third parties and/or the charging infrastructure and /or the vehicle while charging, etc.?

4. Conclusion:

Numerous publications stress the fact that smart grids are the natural complement to electromobility... or the reverse. However, while many technical solutions are now available to facilitate these complementarities, we have shown that some crucial questions about the definition of requirements need to be solved to ensure an efficient and equitable working of those complex systems. A failure to do this would lead to a non-satisfactory collective solution, potentially counter-balancing any positive impact expected by the public concerning smart grids and electromobility. Integrating renewable energies in smart grids to enable a clean mobility needs the technical solutions to be doubled by careful system design based on state-of-the-art requirements work. This challenge is not for within ten years, it is in front of us right now. Given the importance of the results, the size and complexity of this challenge, it deserves the attention of the best of research and industry to tackle it right now.

5. References:

[1]   "A European Strategy for Clean and Energy Efficient Vehicles." European Comission, 2010.

[2]   A. Vidigal, "Renewable energies and smartgrids," in *2011 IEEE EUROCON - International Conference on Computer as a Tool (EUROCON)*, 2011, pp. 1-1.

[3]   Better Place, "From Cars 1.0 to 2.0: An Economic and Environmental Blueprint for the Future of Energy and Transportation." 2008.

[4]   Commission de Régulation de l'Electricité et du GAZ (CREG, "ETUDE relative à l'impact possible de la voiture électrique sur le système électrique belge." 2010.

[5]   H. Roth and B. Gohla-Neudecker, "Analysis of Renewable Energy Power Demand for Specifically Charging EVs," 2009.

[6]   "PlugShare - Electric Vehicle Charging Network." [Online]. Available: http://www.plugshare.com/. [Accessed: 25-Jan-2012].

[7]   "Plugsurfing - The EV-Charging Community." [Online]. Available: http://fr.plugsurfing.org/. [Accessed: 25-Jan-2012].

[8]   "OnStar, Verizon Unveil Second-Generation Research Vehicle to Demonstrate the Power of the Verizon 4G LTE Network," *MarketWatch*.

[9]   "Deloitte Survey: Gen Y's Embrace of Hybrid Vehicles May be Auto Market's Tipping Point." [Online]. Available: http://www.prnewswire.com/news-releases/deloitte-survey-gen-ys-embrace-of-hybrid-vehicles-may-be-auto-markets-tipping-point-137666268.html. [Accessed: 27-Jan-2012].

[10]   C. Fest, "The energy supplier perspective and the integration in the European Market," presented at the Electric Vehicles Conference, Brussels, 2012, p. 27.

[11]   "Smart Grid Information Clearinghouse (SGIC)." [Online]. Available: http://www.sgiclearinghouse.org/ConceptualModel. [Accessed: 25-Jan-2012].

[12]   TEN-T Executive Agency, "Greening European Transportation Infrastructure for Electric Vehicles." [Online]. Available: http://tentea.ec.europa.eu/en/ten-t_projects/ten-t_projects_by_country/multi_country/2010-eu-91117-p.htm. [Accessed: 25-Jan-2012].

3 Requirements Engineering Efficiency Workshop (REEW)

Editors

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

Nazim H. Madhavji
University of Western Ontario, Canada, madhavji@gmail.com

Workshop Programme

Introduction to the 2nd International Requirements Engineering Efficiency Workshop (REEW 2012) at REFSQ 2012

Norbert Seyff[1] and Nazim H. Madhavji[2]
REEW 2012 Co-Chairs

[1] University of Zurich, Switzerland
seyff@ifi.uzh.ch
[2] University of Western Ontario, Canada
madhavji@gmail.com

Preface

Requirements engineering research has focused on specification quality for a long time, leading to recommendations of how to engineer high quality requirements specifications. Practitioners, however, do not have the time and resources for developing theoretically best requirements. Rather, many situations call for short-cuts that allow investing effort in those concerns that are critical for success, while reducing effort in other areas where risk is relatively small. The social context, smart collaboration processes, and novel ways of looking at the interface between stakeholders and the supplier can be a basis for increasing the yield and quality of requirements, while reducing effort.

The International Requirements Engineering Efficiency Workshop (REEW 2012) aims at initiating, facilitating, and nurturing the discussion on efficient approaches to engineer fitting requirements. Requirements engineering is here seen as a means that can be simplified, automated, or combined with other practices to achieve successful systems in an economically efficient manner. REEW 2012 provides a platform to the community of practitioners and researchers that are interested in efficient and pragmatic approaches to requirements engineering.

This volume contains papers accepted for presentation at REEW 2012. Three program committee (PC) members reviewed each paper, and so we are grateful for the time and effort all the PC members, listed below, have generously given to REEW 2012. A motivational talk from the trenches of requirements engineering, the presentation and discussion of the 5 accepted papers, the interactive session on research challenges on requirements efficiency and, of course, the workshop participants characterize the REEW 2012 workshop.

Program Committee

Steffan Biffl, Technische Universität Wien (Austria)
Oliver Creighton, Siemens AG Corporate Technology (Germany)
Maya Daneva, University of Twente (Netherlands)
Jörg Dörr, Fraunhofer IESE (Germany)
Remo Ferrari, Siemens (USA)
Vicenzo Gervasi, University of Pisa (Italy)
Tony Gorschek, Blekinge Institute of Technology (Sweden)
Paul Grünbacher, Johannes Kepler University (Austria)
Andrea Herrmann, Infoman AG (Germany)
Sven Krause, Zühlke Managements Consultants AG (Switzerland)
Soo Ling Lim, University College London (UK)
Andriy Miranskyy, IBM (Canada)
Anna Perini, Fundazione Bruno Kessler (Italy)
Juha Savolainen, Danfoss Power Electronics A/S (Denmark)
Kurt Schneider, Leibniz University Hannover (Germany)

Towards the Use of Software Requirement Patterns for Legal Requirements

Axel Hoffmann[1], Thomas Schulz[2], Holger Hoffmann[1], Silke Jandt[2], Alexander Roßnagel[2], and Jan Marco Leimeister[1]

[1]Information Systems, Kassel University, Germany
{axel.hoffmann, holger.hoffmann, leimeister}@uni-kassel.de
[2]Public Law particulary Environmental Law and Technology Law, Kassel University, Germany
{t.schulz, s.jandt, a.rossnagel}@uni-kassel.de

Abstract. Laws and regulations play an increasingly important role for requirements engineering and system development. The challenge of interpreting the law to elicit legal requirements for a novel application calls for legal expertise. In this paper, we investigate if the effort of compiling a list of legal software requirements can be reduced by reusing recurring legal requirements. Therefore, we collected legal requirements that are stable concerning changes due to their origin in fundamental, higher-ranked laws, and derived software requirement patterns from them. This paper contributes by presenting those software requirement patterns consisting of the name, the goal and the pre-defined requirement template. We argue that under certain circumstances they can be used as a lightweight approach to specify legal requirements in system development projects and hence reduce the need for legal advice.

Keywords: Software Requirement Patterns, Requirements Reuse, Legal Requirements, Laws, Regulations

1   Introduction

The need for system developers to create systems compliant to legislature has been identified as a challenging and important problem in the requirements engineering (RE) community [1, 2]. This trend can be seen, for example, in the finance and healthcare domain, but is also getting more important in other domains of system development [3]. During the design of information systems in particular, one needs to consider: the EU Data Protection Directive, the basic rights to informational self-determination, confidentiality and integrity of information technology systems, the secrecy of telecommunications, as well as the data and consumer protection law. Infringement of any such laws and regulations can lead to high costs, e. g., in the form of compensations or penalties. These litigation-related costs are rising faster than the costs covering all the other aspects of software development; they even outgrow the cost for programming [4]. Only considering laws and regulations and complying with them enables legitimate information system development [5].

Developing legally compliant systems is very challenging. Laws and regulations contain numerous ambiguities, cross-references, and domain specific definitions. Furthermore, they are frequently amended via new regulations and judicature. Although the access to laws and regulations has become easier for system developers in the age of the Internet [2], the problem of the complexity of applying laws is not resolved. Even the identification of relevant laws, and especially the derivation of requirements for the technical system from laws, can hardly be accomplished without legal expertise. Despite the knowledge of specific legal terms and legal reference techniques [6], requirements analysts need to recognize the correlation between the different rules, as well as comprehend the statements of laws relating to technology. Thus, the challenge is already in the development process of interpreting the law and deriving system requirements from them.

Researchers are providing engineers with techniques and tools for specifying and managing software requirements for legally compliant systems [2]. However, these techniques are very laborious and require experience with laws and legal texts. Only a few requirements analysts have such legal expertise. Further, many system development projects cannot afford a comprehensive legal requirements analysis.

The purpose of our research is to help requirements analysts in specifying legal requirements (LRs). We thus compare the results of LRs specifications and derive software requirement patterns (SRPs) [7] that can be (re)used by requirements analysts in system specification. The LRs specifications we used as source material were derived by legal experts with KORA, a method used in German legal research. The acronym KORA stands for "**Ko**nkretisierung **r**echtlicher **A**nforderungen*"* (concretization of LRs) [8], and denotes a procedural method which allows the consideration of LRs in the design of information technology. This method has been evaluated several times in legal research [9-14] and derives requirements from the (stable) purpose of law, rather than handling detailed (changing) regulations. In our study, we have chosen the legal purpose of personal data protection and have derived six SRPs supporting it. These SRPs cannot replace LRs analysis in law-critical domains, but they can serve as a lightweight approach to consider legal purposes in RE.

The remainder of the paper is organized as follows. We first give an overview of the related work with LRs and SRPs. Next, we briefly describe KORA to show why the results are appropriate to create reusable SRPs. After a description of the research design in section 4, we present six SRPs for LRs in section 5. This is followed by the discussion and conclusion.

2 Related Work

2.1 Specifics of Legal Requirements

Laws are normative provisions that describe what is forbidden or allowed. The way in which laws are formulated differs fundamentally from the way in which requirements are specified [15]. As developers of technical systems usually have no legal training, specialists need to be incorporated into the development process to analyze

LRs of law [1]. In determining the LRs for a technology, there are the following basic challenges [1]:

— Choice of laws
— Extraction of relevant obligations and rights from laws
— Abstract and technology neutrality laws
— Dynamics of law

Due to the large number of laws, it is hard to assess which laws, along with their LRs, need to be considered for the development of a specific information system. Given today's global distribution of technical systems, laws of different countries can be relevant. Additionally, there is a prevailing legislative hierarchy: in Germany, for example, the constitution, laws at the federal and state level, and regulations. Thus, the developer is faced with a multitude of laws, some of which are parallel, but which may also occur secondarily [1, 2]. The legal analysis is complicated by the fact that not only the (written) laws, but the interpretation by the courts, as well as that in the literature, must also be taken into account [2]. These are harder to obtain than legislation, and can sometimes produce a more mixed picture, with decisions regarding specific cases taken independently. It is precisely challenging for this reason to identify relevant sources of LRs.

After the relevant LRs have been identified, the next challenge awaits. From the often very long laws or legal interpretations, relevant rights and obligations need to be extracted in order to provide LRs. It is common for technical systems that only a small part of existing legislation is relevant. In addition, dynamic and static references in the laws make the related interpretation more difficult [2, 16].

Laws set particular legal consequences for an unlimited number of individual cases, and must generally be formulated abstractly. This requires laws to be interpreted before they can be applied to specific cases. Further, laws often provide a margin of their interpretation, since names and phrases can be ambiguous [2]. In RE, this is referred to as a defect of natural language. In legal literature, interpretations can be found that do not meet the intent of the laws. Additionally, there are often varying legal views [2]. Laws usually address issues that have occurred in the past, such as problems that were caused by economic or social changes. For advanced information systems, the relevant specific details in laws are missing because economic or social changes have not yet taken place in practice, and legislature has not yet intervened [6]. Moreover, it is not possible for the legislature to adapt the laws at the same pace as that which technology development moves. This issue is largely met with abstract and technology-neutral regulations that target only specific risks. However, it is possible that regulations are missing for certain legal risks of new technology [1].

Laws are not necessarily time-consistent and changeless; rather, they are subject to continuous changes [2]. Especially laws and detailed regulations at the lower levels of hierarchy may change quickly, or are supplemented by additional regulations. Further, interpretation of laws by judgments is often necessary for the sake of legal security, but this is a very time consuming process [17]. Thus, for advanced information systems, concrete points of reference may possibly not yet be available. Compounding matters, the interpretation of laws can change over time [18], and the law dynamics

require technical system dynamics. These need to be adjusted throughout the life cycle with respect to changing or new laws [1]. This traceability of requirements raises a further problem area in RE. If the system has to be adapted, it needs to be documented which design decision is influenced by which (legal) requirement.

2.2 Prior Work on Legal Requirements

In software engineering, different efforts have been made to deal with LRs. An indepth survey of work within the computer science and artificial intelligence domains in handling legal texts for system development has been carried out by Otto and Anton [2] to aid requirements analysts to better specify, monitor, and test information systems for compliance. This section provides a brief overview of approaches that help requirements analysts in acquisition and analysis of LRs.

Siena et al. [15] recommend the transition of LRs into stakeholder goals, and that they should be considered in goal-oriented RE. The described approach corresponds with the explanations of Ishikawa et al. [17], in which they stress the transition between legal goals and the stepwise refinement of technical goals. As laws are often very abstract and general, it is essential for a business organization to derive its own concrete measures to be taken. However, these legal regulations do not comprise the goals of RE; rather, they equal the concept definitions that require further refinement. As described by Ishikawa et al. [17], goal refinement and the refinement of concept definitions are related to each other. Guarda and Zannone [19] deal with LRs in a goal-oriented way, as they derive goals directly from law and consider them in the later requirements analysis. Problems arise when there are no laws or regulations that can be interpreted and used directly by requirements analysts.

Moreover, there are approaches that translate laws into abstract models [6]. It is therefore possible to formally examine an application in terms of legal conformity. However, this translation of requirements into abstract models requires an exact formulation that regulations often lack, as they are in many cases too general and nontechnical [2]. Even if these regulations were to offer a sufficient level of accuracy, there would still be the complexity of translating abstract legal concepts into requirements [15]. Methods for the interpretation of these regulations are not sufficiently advanced, concentrating more on specific aspects [1]. Thus, only explicit guidelines allow applying requirements modeling to legal regulations in order to obtain requirements for the system. Abstract laws need to be concretized in advance.

Toval et al. [5] have set up a LR catalog regarding security and personal data protection which serves as a source of documents and interpretations for system development teams. The catalog enables requirements analysts to incorporate LRs into specifications, and thus build compliance into new systems. This approach, however, still faces the problem of dynamics in legislation and associated changes [2].

2.3 Requirements Reuse and Software Requirement Patterns

Reuse is an established practice in software engineering [20, 21]. In RE, reuse can help requirements analysts to elicit and document software requirements. SRPs are a

worthwhile approach to reuse requirements [22]. A pattern, in general, describes a problem which occurs over and over again, and then describes the core of the solution to that problem, in such a way that it can be used a million times over, without ever doing it the same way twice [23]. SRPs are used for the software analyses stage. There are different approaches that differ in scope, notation and application [22]. Recent approaches using SRPs for writing software requirement specifications can be found in the work of Withall [7] and in the Pattern-based Requirements Elicitation (PABRE) by Renault, Mendez-Bonilla, Franch, and Quer [24, 25].

A pattern based approach can reduce the effort of acquiring requirements for many development projects. The possible benefits for requirements analysts are not only the reduction of time spent to perform the elicitation of the requirements, but also the improvement of the quality of the requirements book obtained [25]. For this reason, the reusability of SRPs is the prerequisite for their applicability in practice.

Summarizing, the challenges with LRs analysis evident from: choice of laws, extraction of relevant obligations and rights from laws, abstract and technology neutrality laws, and dynamics of law demand specific knowledge and considerable effort in RE. We seize the suggestion of LR reuse [5] and implement it with SRPs [7, 24] to face the named challenges. With the use of LRs that are stable concerning changes in detailed regulations due to their origin in fundamental, higher-ranked laws, we reduce flaws existing in prior LR reuse. In order to generate SRPs, we use specifications containing LRs homogeneously created with the KORA-Method.

3   KORA – Concretization of Legal Requirements

KORA is a method that has been used in German legal research to derive LRs for technical systems for nearly 20 years [8-14]. KORA is performed by legal experts and is not meant to be performed by requirements analysts. Nevertheless, we used the results of various applications of KORA to identify SRPs for LRs. For ease of understanding, we briefly describe the basics of KORA (with the specific terminology) in the following section.

3.1   Deriving Legal Requirements from Higher-Ranked Laws

For the consideration of legality of systems in computer science, the concept of IT compliance has been established. To this end, laws are analyzed for containing direct or indirect LRs - a step which must be considered in the design of technology. Examples are the Digital Signature Act and the Data Protection Act. From them, legally binding technical requirements can be obtained directly, as failure of implementation could result in legal consequences. For this circumstance, the understanding of laws and other LRs as constraints has emerged.

The minimum requirements for a socially responsible technology design can be found in the law. These serve both the constitutionally guaranteed free democratic basic order of the state and the protection of fundamental rights of individual citizens. Some laws, such as the data protection legislation, contain explicit guidelines for the

design of data processing information systems. In addition, there are design requirements in other laws that regulate only indirect information technology, such as in accordance with § 312g of the German Civil Code (BGB), regarding entrepreneurs fulfilling legal duties in the electronic exchange. Therefore, KORA obtains technical requirements from the purpose of legislation [8, 9]. This is called being legally compatible. For the purpose of the secrecy of telecommunications, e. g., a communications technology where communication is encrypted automatically is more legally compatible than one that is not automatically encrypted; albeit, the unencrypted technology is not in any case unlawful. Further, by permanently validating laws and their purposes, it is not necessary to adapt the systems as a result of legislative changes. At the time of development, loopholes in detailed rules are also irrelevant [26].

In the development of technical systems - similar to the task of a judge in determining the facts of the case - developers have to derive specific technical requirements from the legal provisions. However, this task has to be carried out before there is a finished information system. With KORA, the legal concretization is achieved through a four-step process (Fig. 1) [12].
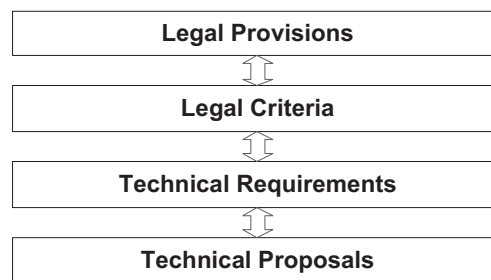
Fig. 1. Levels of KORA

3.2   Application of KORA in Legal Research

KORA starts from existing constitutional and other legal norms, which can be specific legal rules. If there are no specific legal provisions applicable to the planned information system, or if they are subject to short-term changes, KORA starts from steady higher-ranked legal rules, such as can be found, for example, in the constitution [9, 12]. On the basis of the purpose and the knowledge of social chances and risks inherent in the information systems, legal provisions for the planned information system are developed from the constitutional and other legal norms on the first level. Hence, the legal provisions apply to the specific project. By focusing on higher-ranked legal rules, the number of laws to be examined is narrowed down, which simplifies the selection of relevant laws [26]. Furthermore, the differences between the laws to be considered in different jurisdictions are far greater on the lower-ranking level. If an information system is used worldwide, it must indispensably be aligned with general provisions.

Legal criteria are identified by analyzing how the legal provisions that have been developed on the first level can be qualitatively assessed with regard to the information system [9]. The criteria describe rather abstract solutions to fulfill the legal provisions which are in principle legal and non-technical, but certainly can be technical. Legal criteria can also be developed on the basis of the reasoning given by judges in legal cases in which the same legal norms are applied [9]. Sometimes the criteria can already be incorporated as design demands in detailed legislature.

Technical requirements for the design of the technology are abstractions of specific characteristics of the technology. As the objective of KORA is not only a lawful but also a legally compatible design of information systems, the technical requirements for design are requirements which can enhance the legal compatibility of information systems. A high degree of legal compatibility ensures sustainable lawfulness and lawfulness in different jurisdictions [26]. If they are adopted in the system development, there will still remain considerable scope for the implementation by designers. For complex systems, further technical concretization should take place afterwards.

On the last level of KORA, technical proposals for the design of the technology are developed on the basis of the technical requirements [9]. Technical proposals for the design are performance characteristics which constitute technical functions. For a new information system, technical characteristics are developed from the technical requirements for the design.

3.3 KORA-Results as Foundation of Software Requirement Patterns

We argue that the results of KORA are suitable to be used as source of SRPs. Due to their origin in general and stable legal rules, they are most suited for requirement reuse because it ensures a long life period of the SRPs. Since these rules are often recognized internationally, the patterns can often also be used for systems that are intended for an international market. Thereby one should orientate by the strictest rules if possible. By using German law for privacy purposes, there is a high likelihood that the information system is legally compliant with other jurisdictions. Further, due to the focus on legally compatible systems rather than just archiving the minimum standards of law, it ensures legal compliance even when detailed laws are tightened.

KORA results in requirements on different levels of abstraction. Technical proposals (level 4) provide design recommendations for the technical system. Nevertheless, for SRPs, we need solution-free requirements [27] that can be found in the technical requirements (level 3), which are related to the basic functionality of the information system. We extract the technical requirements from the LR specifications and use them as source requirements to generate SRPs.

4   Research Design

Results from LR specifications, all which were archived with KORA, served as our source material. Some of the documents were available in public [11-14], while others were not designated for public use, but were provided for our research.

Given the documents, we followed the systematic approach of Withall [7] to find candidates for SRPs, and scanned a sample of seven LR documents. These documents contained about 30 to 50 LRs. We listed all requirements in a spreadsheet. If a requirement was similar to one we already had on the list, we noted that and moved on. In the end, we filtered the list for all requirements that were mentioned in more than one specification. For the identified recurring requirements we formulated SRPs.

5   Results

With today's technology, it is especially the protection of personal data that often plays an outstanding role. Accordingly, this paper focuses on SRPs which are particularly relevant for the protection of personal data. These patterns are not exhaustive, and should serve only as examples to illustrate reusable SRPs.

We have selected natural language to formulate the SRPs. Non-technical experts, such as legal practitioners, prefer natural language requirements for reading, analysis and discussion [5]. However, the software requirements specifications we used as a source were also written in natural language. This is in line with recent approaches using SRPs for writing software requirements specifications [7, 24].

To illustrate the pattern for LRs, we use the following attributes that are components of the recommended structure of a SRP in [22]:

- Goal: The goal has the role of the problem part of a pattern. It has an important role since it will help to decide whether the pattern is applicable to the software [25]. This is determined by the planned functionality of the software.
- (Fixed Part) Template: The fixed part template is the core of the solution, stating that the software has to achieve the goal of the SRP, but not indicating how this goal can be achieved. Since the fixed part of a form is abstract, it is possible to provide some extra-information or constraints in the extension part about how to achieve the goal of the SRP [25].
- Sources: The sources usually comprise the source documents. For our purposes, we provide the legal provisions from which the requirements were derived and cite LRs specifications in which the derivation is described.

The example patterns are ascribed to the informational self-determination [11-14]. The right to informational self-determination is a special manifestation of the right of development and protection of one's personality, which is established in Art. 2 (1) read in conjunction with Art. 1 (1) of the German Constitution. This right was acknowledged in 1983 by the German Federal Constitutional Court [28]. By the right to informational self-determination, the individual is protected from unlimited dealing with personal data. Individuals need to decide for themselves when, and within which limits, personal life issues should be revealed. Today, the right to informational self-determination has a big impact, especially on the data protection acts.

The following are a few examples of SRPs that are particularly relevant for the protection of personal data (Table 1).

Table 1. Software Requirement Patterns

1	**Confidentiality of the Communication Channels**	
	Goal	Ensure protection and confidentiality of personal data during transmission.
	Template	The system shall prevent spying out personal data by unauthorized third parties during transmission.
	Source	[28]; derived in, e.g., [11-14].
2	**Divide of Different Personal Data**	
	Goal	Limit the usage of personal data to the dedicated purpose.
	Template	The system shall divide personal data according to different purposes and coherences of use.
	Source	[28]; derived in, e.g., [11, 12].
3	**Control about Storage Medium**	
	Goal	Ensure protection and confidentiality of personal data during storage.
	Template	The system shall store personal data on a storage medium that is exclusively controlled by the user.
	Source	[28]; derived in, e.g., [11-13].
4	**Access Control**	
	Goal	Ensure protection and confidentiality of personal data during storage.
	Template	The system shall ensure that only authorized users gain access to the service.
	Source	[28]; derived in, e.g., [11-13].
5	**Limitation of Storage Time**	
	Goal	Limit the usage of personal data to the dedicated purpose.
	Template	The system shall delete personal data if they are no longer necessary for system operations.
	Source	[28]; derived in, e.g., [11, 12].
6	**Documentation of Processing with Personal Data**	
	Goal	Ensure transparency of personal data usage.
	Template	The system shall record processing with personal data.
	Source	[28]; derived in, e.g. [12].

6   Discussion

There are many pitfalls when formulating legal SRPs in order to ensure the applicability of the result in more than just one system development project; fortunately, there are also some advantages. Legal SRPs satisfy the need of requirements analysts in three situations. First, they can help if no detailed laws or regulations are applicable. Second, they are very useful if the requirements analysts do not have any exper-

tise to work with laws and regulations. Third, they are essential if there are too few resources to conduct a comprehensive LRs analysis.

To reduce the disadvantages of LRs reuse, we considered specifics in LRs engineering. Usually, a LR catalog requires updates each time the law changes. This is also true for legal SRPs. But with the selection of requirements worked out with the KORA-Method, we take advantage of the specifics. The KORA-Method that derives the requirements from general and stable legal rules ensures a long life period of the SRPs even without permanent updates. Further, due to the focus on legally compatible systems rather than just archiving the minimum standards of law, it ensures in all likelihood legal compliance even when detailed laws are tightened.

The traceability between the derived requirements and the sources in law are ensured by specifying the legal sources mentioned in the analyzed source specifications. Further, the full trace from the LRs to the legal sources can be found, if necessary, in the KORA specifications.

When a pattern is to be used, it first has to be examined whether this pattern is relevant for the design of the information system at all [24]. If, for example, a system does not gather, process or utilize personal data, a pattern which only purposes the protection of such data must not be adopted. After identifying all relevant SRPs, the requirements analyst can assemble the requirements document [24].

The effort for selecting and adapting SRPs is much less than a full requirements analysis. According to the domain, while the search, extraction and translation of regulations into requirements took up to several weeks, the selection and adaption of SRPs can be done in four to five man-days [25].

With LRs, there is always the problem that the legislature can change them at any time. For patterns which are deduced from LRs, this means that they can be deprived of their legal basis. This is especially a problem in dealing with relatively detailed laws, since these can change frequently. Fundamental legal provisions, however, remain very stable. Patterns are therefore more stable when they are deduced from more stable law. For the use of patterns in practice, their stability is very important. For this reason, we developed patterns which can be ascribed to fundamental, higher-ranked laws.

For use in practice, it is also important that the patterns are reusable. Only in this way is the considerable effort to create patterns worth. To ensure the reusability, we developed patterns by means of technical requirements derived in different projects for different systems. A further challenge in the development of such patterns is that they implement legal provisions, but should be used by engineers. This assumes that the patterns are formulated in a language that can be understood by engineers. For this reason, our patterns were formulated in technical language. It could thus be ensured that there were no misunderstandings with the use of patterns due to linguistic differences between the legal and technical languages.

Not the least of the challenges, the patterns must also be legally correct. Patterns which should implement legal provision should be evaluated with the cooperation of jurists; accordingly, jurists were involved in the design and evaluation of each pattern. Thus, the derived SRPs are conform to today's detailed laws.

7   Conclusion

The fulfillment of LRs cannot be reached by supplementing individual software components or modules to a system, as they affect the whole software, compared to a cross-cutting concern of aspect-oriented programming [17]. LRs resulting from the laws must therefore be considered in the early phases of RE in order that the legally compliant technology design can be ensured at early stages of development [15]. Early consideration of LRs does not take place in most current development projects. For example, requirements of informational self-determination which have already been established by comprehensive data protection legislation are as important as functional requirements when designing information systems; however, they are not elicited, analyzed and taken systematically into account during implementation [19].

SRPs offer a solution for requirements analysts to factor LRs directly into the information system design. These patterns are generalizable, which leads to reusability. We created the patterns from LRs that were deduced from stable higher-ranked laws, resulting in the development of stable patterns. Additionally, we formulated the patterns in a technical language to guarantee that even requirements analysts without a legal background could work with them. With our patterns, requirements analysts have a lightweight approach to incorporate LRs into system specifications. It can improve the productivity of requirements analysts, as they can start from a set of predefined SRPs in a technical language. The quality of the specification can also be enhanced because the SRPs are evaluated by legal experts.

Our future plan is to integrate the requirement patterns within a SRP catalog. Further, we want to parameterize some parts to allow more detailed choices by each analyst applying the pattern and make it easier to adapt the patterns.

8   References

1. Kiyavitskaya, N., Krausova, A., Zannone, N.: Why Eliciting and Managing Legal Requirements Is Hard. In: Requirements Engineering and Law, pp. 26-30. (2008)
2. Otto, P.N., Anton, A.I.: Addressing Legal Requirements in Requirements Engineering. In: 15th IEEE International Requirements Engineering Conference, pp. 5-14. (2007)
3. Hoffmann, A., Söllner, M., Fehr, A., Hoffmann, H., Leimeister, J.M.: Towards an Approach for Developing socio-technical Ubiquitous Computing Applications. In: Sozio-technisches Systemdesign im Zeitalter des Ubiquitous Computing, Berlin (2011)
4. Cosgrove, J.: Software engineering and the law. Software, IEEE 18, 14-16 (2001)
5. Toval, A., Olmos, A., Piattini, M.: Legal requirements reuse: a critical success factor for requirements quality and personal data protection. In: 10th IEEE International Requirements Engineering Conference, pp. 95-103. (2002)
6. Breaux, T.D., Anton, A.I., Boucher, K., Dorfman, M.: Legal Requirements, Compliance and Practice: An Industry Case Study in Accessibility. In: 16th IEEE International Requirements Engineering Conference, pp. 43-52. (2008)
7. Withall, S.: Software requirements patterns. Barnes & Noble (2008)
8. Hammer, V., Pordesch, U., Roßnagel, A.: KORA–Eine Methode zur Konkretisierung rechtlicher Anforderungen zu technischen Gestaltungsvorschlägen für Informations-und Kommunikationssysteme. Infotech/I+ G 21–24 (1993)

9. Hammer, V., Pordesch, U., Roßnagel, A.: Betriebliche Telefon- und ISDN-Anlagen rechtsgemäß gestaltet. Springer, Berlin (1993)
10. Jandt, S.: Vertrauen im Mobile Commerce–Vorschläge für die rechtsverträgliche Gestaltung von Location Based Services. Baden-Baden (2008)
11. Gitter, R.: Softwareagenten im elektronischen Geschäftsverkehr – Rechtliche Vorgaben und Gestaltungsvorschläge, Baden-Baden (2007)
12. Steidle, R.: Multimedia-Assistenten im Betrieb – Datenschutzrechtliche Anforderungen, rechtliche Regelungs- und technische Gestaltungsvorschläge für mobile Agentensysteme, Wiesbaden (2005)
13. Ranke, J.S.: M-Commerce und seine rechtsadäquate Gestaltung – Vorschläge für vertrauenswürdige mobile Kommunikationsnetze und -dienste, Baden-Baden (2004)
14. Idecke-Lux, S.: Der Einsatz von multimedialen Dokumenten bei der Genehmigung von neuen Anlagen nach dem Bundesimmissionsschutz-Gesetz, Baden-Baden (2000)
15. Siena, A., Mylopoulos, J., Perini, A., Susi, A.: From Laws to Requirements. In: Requirements Engineering and Law, pp. 6-10. (2008)
16. Maxwell, J.C., Antón, A.I., Swire, P.: A Legal Cross-References Taxonomy for Identifying Conflicting Software Requirements. In: 19th IEEE International Requirement Engineering Conference. (2011)
17. Ishikawa, F., Inoue, R., Honiden, S.: Modeling, Analyzing and Weaving Legal Interpretations in Goal-Oriented Requirements Engineering. In: Requirements Engineering and Law, pp. 39-44. (2009)
18. Massey, A.K., Otto, P.N., Anton, A.I.: Prioritizing Legal Requirements. In: Requirements Engineering and Law, pp. 27-32. (2009)
19. Guarda, P., Zannone, N.: Towards the development of privacy-aware systems. Information and Software Technology 51, 337-350 (2009)
20. Berkovich, M., Esch, S., Leimeister, J.M., Krcmar, H.: Requirements engineering for hybrid products as bundles of hardware, software and service elements – a literature review. 9. Internationale Tagung Wirtschaftsinformatik (WI 2009), Wien, Österreich (2009)
21. Berkovich, M., Leimeister, J., Krcmar, H.: Requirements Engineering for Product Service Systems. Business & Information Systems Engineering 3, 369-380 (2011)
22. Franch, X., Palomares, C., Quer, C., Renault, S., De Lazzer, F.: A Metamodel for Software Requirement Patterns. Requirements Engineering: Foundation for Software Quality 85-90 (2010)
23. Alexander, C.: The timeless way of building. Oxford University Press, USA (1979)
24. Renault, S., Mendez-Bonilla, O., Franch, X., Quer, C.: PABRE: Pattern-based Requirements Elicitation. In: Research Challenges in Information Science, 2009. RCIS 2009. Third International Conference on, pp. 81-92. (2009)
25. Renault, S., Mendez-Bonilla, O., Franch, X., Quer, C.: A Pattern-based Method for building Requirements Documents in Call-for-tender Processes. International Journal of Computer Science and Applications 6, 175 - 202 (2009)
26. Hoffmann, A., Jandt, S., Hoffmann, H., Leimeister, J.M.: Integration rechtlicher Anforderungen an soziotechnische Systeme in frühe Phasen der Systementwicklung. In: Mobile und ubiquitäre Informationssysteme, Kaiserslautern (2011)
27. Firesmith, D.G.: Engineering security requirements. Journal of Object Technology 2, 53-68 (2003)
28. BVerfGE (anthology of the judicial decisions of the German Federal Constitutional Court): 65, 1 - court ruling of the 15th December, 1983 - 1 BvR 209/83 et al., (1983)

Enhancing Requirements Engineering Efficiency Using Explicit Semantics and Template-Based Mechanisms
Research Preview

Thomas Moser[1], Wikan Sunindyo[1], Stefan Farfeleder[2], Inah Omoronyia[3]

[1] Christian Doppler Laboratory CDL-Flex, Vienna University of Technology, Austria
{moser, wikan}@ifs.tuwien.ac.at
[2] Institute of Computer Languages, Vienna University of Technology, Austria
stefanf@complang.tuwien.ac.at
[3] The Irish Software Engineering Research Centre, University of Limerick, Ireland
inah.omoronyia@lero.ie

Abstract. Ontologies are used to support a range of requirements engineering (RE) tasks, including the elicitation and analysis of requirements. Major challenge in RE are the efficient handling of requirements consistency, completeness and maintainability. Typically, RE tasks based on explicit semantics serve separate purposes and therefore do not address overall RE efficiency. An open issue is how different ontology-based approaches used in RE can be combined providing a beneficial synergy of these approaches. In this paper we propose to integrate two separate approaches building upon requirements templates and ontologies, one guiding requirements elicitation using Boilerplates, the other one performing requirement conflict analysis using EBNF. We present an evaluation concept to empirically evaluate the synergy benefits and efforts of integration based on a real-world industry study. Expected results are that this integration approach can help improving the overall RE efficiency.

Keywords: requirement elicitation, requirements engineering efficiency, conflict analysis, requirements categorization, ontology, requirement template.

1   Introduction

Modern software and systems engineering projects are challenging, in part, due to the high number and complexity of requirements. Further, geographically distributed project stakeholders usually have diverse backgrounds and sometimes even use different domain terminologies [5]. Therefore, a major goal and challenge of requirements engineering (RE) is to achieve consistent requirements descriptions in order to create a common and agreed understanding on the set of requirements between all project stakeholders. Semantic technologies seem to be a promising approach to address these challenges. Ontologies provide the means for describing the concepts of a domain and the relationships between these concepts in an explicit and machine-understandable way allowing automated processing and inference of the available

information [4]. Several ontology approaches [1, 4] have been used to support requirements engineering, such as for guiding requirements elicitation, for requirements conflict analysis or for requirements categorization. However, these approaches are still separately used and have not explored possible synergies of different approaches.

In this paper, we provide a methodology that is capable of integrating different ontology-based RE approaches. As proof-of-concept, we integrate two RE methods using explicit semantics, named ontology-based requirements elicitation and ontology-based requirements categorization. The requirements elicitation tool DODT [2, 8] transforms natural language requirements into a corresponding semiformal linguistic template representation, also known as boilerplates. In addition, OntRep [6, 7] provides an automated ontology-based reporting approach for requirements categorization, conflict analysis, and tracing based on ontologies. The objective of this research is to provide the benefits of both approaches during requirements engineering, thus showing the benefits of increasing efficiency of requirements engineering with explicit semantics. Basis for this integration is the transformation between the used two requirements templates, EBNF and boilerplates. We discuss an evaluation concept for empirically evaluating the benefits and effort of integrating both approaches using real-world industrial requirements from the automotive domain. As evaluation criteria for the RE efficiency, we plan to measure the effort for managing requirement consistency, completeness and maintainability, and therefore additionally put the focus of our evaluation on the overall visibility and quality improvement in RE.

The remainder of this paper is structured as follows: Section II summarizes related work on ontologies for RE; Section III presents the solution approach and finally Section IV presents the evaluation concept and expected results.

2 Related Work

The use of ontologies for addressing requirements elicitation problems was proposed by Kaiya and Saeki [4]. They were motivated by findings that the lack of domain knowledge during requirements elicitation resulted in low quality specifications. They subsequently use domain ontologies as storage for domain knowledge to support requirements elicitation. However, the experiment using a case study of software music players was too small to argue for sufficient generality of the experimental findings.

Dzung and Ohnishi [1] propose a requirements ontology for requirements elicitation. Their proposed requirements ontology represents (1) a functional hierarchy of a certain software system, (2) relationships among functional requirements, and (3) attributes of functional requirements. By using this ontology, they measured the correctness and completeness of elicited requirements. However, further experimental evaluation seems advisable to strengthen the external validity of the results.

Omoronyia *et al.* and Farfeleder *et al.* [2, 8] proposed the use of ontologies for guiding requirements elicitation. The aim in these related works was to investigate an approach for building domain ontologies from existing technical standards. Omoronyia *et al.* present an evaluation of their approach and provide insights on the challenges of semi-automatically building domain ontologies using natural language

texts. This approach helps reducing the effort of building domain ontologies from scratch. However, further investigation on the possibility of combining this ontology approach seems advisable to provide added benefits for requirement elicitation.

Moser *et al.* [6, 7] use semantic technology for automating the detection of complex semantic conflicts between software requirements. In their work, a semantic approach is used as foundation for automating requirements conflict analysis using the ontology-based reporting tool OntRep. The evaluation was applied to two real-world industrial use cases: (a) different types of conflicts, and (b) different levels of conflict complexity. However, this approach does not use domain ontologies for requirements elicitation. Therefore, synergies with the guidance for domain ontology building approach presented by Omoronyia *et al.* [8] could be beneficial.

Yanhui [9] proposes an ontology integration algorithm as follows: (1) identify alignment between related entities which are semantically correlative, (2) find the places where ontologies overlap and integrate ontologies, (3) prune integrated ontology through detecting ontology redundancy, (4) check the consistency of the integrated ontology. We use this work to design our own ontology integration approach for integrating two different requirements ontologies.

3 Solution Approach

This section presents the integration methodology as solution approach of the planned research. The methodology to integrate different ontology-based RE approaches can be defined as follows: (1) Identify different templates used for requirements elicitation in industrial practice, consider transformation between those templates; (2) analyze domain ontologies used for requirements representation, identify similarities, relationships and conflicts among ontologies; (3) provide integration tools based on analysis results in step 2, integrate different requirements items by using those tools (see Fig.1 for implementation of this methodology).
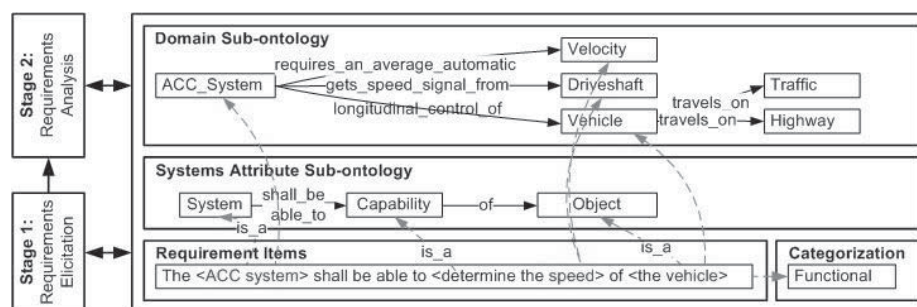
Fig. 1. Interlinking between requirements and ontology representation.

The synergy of the used boilerplate and EBNF grammars has its foundations in the different forms of ontological knowledge representation. The core ontologies used in boilerplate representation include the domain specific ontology, systems attributes ontology and the requirements classification ontology. The domain ontology defines domain specific concepts and the inference rules that describe the axioms, relations

and attributes of these concepts. The systems attribute ontology generally refers to the concepts which, when described properly, can enable the specification of the functional and non-functional characteristics of the system. These are the attributes that are subsequently used to define the structure of a boilerplate grammar.

As shown in Fig.1, we propose to support two major phases of RE, namely Requirements Elicitation and Requirements Analysis. In the elicitation phase, a requirement declaration "*the <ACC system> shall be able to <determine the speed> of <the vehicle>*" conform with the boilerplate templates <System> shall be able to <Capability> of <Object>, where the term *<ACC system>* is linked to the "*System*" concept in the system attribute ontology and also refers to "*ACC System*" as a concept in the domain ontology. Similarly, *<determine the speed>* is linked to the "*capability*" concept in the systems attribute ontology, while the term speed itself refers to a number of concepts in the domain ontology, including *Driveshaft* and *Velocity*. Finally, the requirement statement as a whole is linked to the "*Functional*" concept in the requirements classification ontology. The mapping of a requirement statement to the systems attribute and domain ontology can be achieved using NLP and different similarity measures as demonstrated in previous work [8].

4   Evaluation Plan and Expected Results

As a use case for the evaluation of RE efficiency, we use DODT and OntRep for requirements elicitation and requirement analysis respectively. We choose these tools because we have direct access and experience regarding both tools. DODT focuses on the requirements elicitation, transforming natural language requirements into boilerplate representation, while OntRep is used to categorize the requirements and to identify potential requirement conflicts. Currently, both tools are part of separate RE processes. We expect that by combining the different approaches of both tools we can exploit the advantages of both approaches.

Combining different ontology-based mechanisms to efficiently support requirements engineering stages can improve the quality of the underlying requirements, such as requirements consistency, completeness and maintainability. Furthermore, the usage of explicit semantics for enhancing the presented requirements quality criteria will most likely provide more efficient means than manual approaches or approaches focusing on a single usage only, since artifacts (e.g., domain ontologies) can be reused for a set of approaches. The following paragraphs describe each of these quality criteria and empirical evaluations planned for measuring these quality criteria.

Requirements consistency. To enable us to precisely realize a requirements reasoning engine based on domain ontologies we identified two main sources of inconsistency. These include inconsistency resulting from specific values given to parameters within the system, and *conceptual inconsistency*. The focus of this work is on the latter. Conceptual inconsistency results from the use of conflicting concepts in the achievement of a specified system goal. Conflicting concepts are concepts that will generate requirements inconsistency if the phenomenon within which they are described can result in inappropriate system behavior. For example, the concepts 'door

open' and 'door close' for a railway domain are prone to potential conflicts as a train door cannot be open and closed at the same time. While these are desired properties of a train there is need for careful tradeoffs to be made such that their co-existence is within acceptable risk. The goal of this quality criterion is not to claim that requirements are inconsistent with each other when they reference concepts that have the potential to conflict with each other. But rather we aim to highlight a pointer to possible conflicting and design challenging phenomenon. The requirements analyst or domain expert can then ensure that such requirements are described within acceptable risk and hence avoid an unacceptable behavior of the system.

Requirement completeness. We distinguish between two different kinds of completeness in this research. *Internal requirements completeness* [3] means that individual requirements include the entire information necessary to validate and implement them, e.g., all pre- and post-conditions. On the other hand *external requirements completeness* focuses on the completeness of the overall set of requirements, i.e., that no requirement has been left out and all aspects of the system to be built have been thoroughly specified. The first kind of completeness can be established by using template-based mechanisms for requirements specification. The right kind of patterns ensures that no vital information is being forgotten. This includes specifying events, states and modes for functional requirements and measurement quantities and units for quality requirements. Such patterns can also be easily adapted to additional needs of a domain. The domain ontology information addresses external requirements completeness. Usually a domain ontology encompasses more information than what is actually used in a specific project, i.e., requirements interact with a subset of the entire domain, so simply checking whether all domain terms have been used is not feasible. Instead we need to take the links between ontological concepts into account. If we have the knowledge that a *door* requires to include a *door sensor* in a domain, we should have requirements about the door sensor once we have door requirements. Otherwise it is reasonable to assume that door sensor requirements are missing.

Requirement maintainability. In the scope of this work, we define requirement maintainability as the effort required for performing typical RE maintenance tasks such as requirement categorization or requirements conflict analysis. In a large software project, tasks like requirements categorization, conflict analysis, and tracing require human effort that often prohibits their use in practice. Therefore, software projects often end up with unstructured requirements and conflicts that get discovered late and expensively. In this context, the main research question regarding this requirement quality criterion is: To what extent can a semantic-based approach increase the effectiveness and efficiency of requirements categorization and conflict analysis compared to a traditional manual approach? In order to address the research question we derive the following variables to consider for evaluation: *number of requirements* and *number of requirement categories* used to categorize the requirements. Further, the *total number of true requirements conflicts* existing in a list of requirements, which can be identified by various approaches for conflict detection. Dependent variables that we want to study by the evaluation are: *number of conflicts identified*, *true conflicts that have not been identified* and the *plausibility of requirements classification*. Besides these parameters we also record the effort for requirements categoriza-

tion and conflict analysis. This includes *preparation effort* (e.g., creating the used ontology), *categorization effort*, and *conflict analysis effort*.

Expected result are that the integration approach can help improving the overall RE efficiency by providing better means for handling typical requirements quality criteria such as requirements consistency, completeness and maintainability based on requirements templates and explicit semantics.

Acknowledgments

References

1. Dzung, D.V., Ohnishi, A.: Improvement of Quality of Software Requirements with Requirements Ontology. In: 9th International Conference on Quality Software (QSIC '09), pp. 284-289. (2009)
2. Farfeleder, S., Moser, T., Krall, A., Stålhane, T., Omoronyia, I., Zojer, H.: Ontology-Driven Guidance for Requirements Elicitation. In: 8th Extended Semantic Web Conference, pp. 212-226. (2011)
3. Firesmith, D.: Specifying Good Requirements. Journal of Object Technology 2, 77-87. (2003)
4. Kaiya, H., Saeki, M.: Using Domain Ontology as Domain Knowledge for Requirements Elicitation. In: 14th IEEE International Conference Requirements Engineering, pp. 189-198. (2006)
5. Lamsweerde, A.v.: Goal-oriented requirements engineering: from system objectives to UML models to precise software specifications. Proceedings of the 25th International Conference on Software Engineering, pp. 744-745. IEEE Computer Society, Portland, Oregon. (2003)
6. Moser, T., Winkler, D., Heindl, M., Biffl, S.: Automating the Detection of Complex Semantic Conflicts between Software Requirements: An empirical study on requirements conflict analysis with semantic technology. In: 23rd International Conf on Software Software Engineering and Konwledge Engineering (SEKE 2011), pp. 729-735. Knowledge Systems Institute Graduate School, USA. (2011)
7. Murata, T.: Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE 77, 541-580. (1989)
8. Omoronyia, I., Sindre, G., Stålhane, T., Biffl, S., Moser, T., Sunindyo, W.: A Domain Ontology Building Process for Guiding Requirements Elicitation. In: Wieringa, R., Persson, A. (eds.) Requirements Engineering: Foundation for Software Quality, vol. 6182, pp. 188-202. Springer Berlin / Heidelberg. (2010)
9. Yanhui, L.: An approach to ontologies integration. In: Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2011), pp. 1262-1266. (2011)

Focusing on the "Right" Requirements by Considering Information Needs, Priorities, and Constraints

Sebastian Adam, Norman Riegel, Anne Gross

Fraunhofer IESE, Fraunhofer-Platz 1, 67663 Kaiserslautern
{sebastian.adam, norman.riegel, anne.gross}@iese.fraunhofer.de

Abstract. The trend in industry towards agile and lean approaches requires "good enough" rather than perfect requirements. One means for achieving this aim is to streamline requirements processes by focusing on the right requirements, i.e., on requirements that are economically feasible, most valuable for customers, and relevant for development engineers when making design decisions. In this research preview, we present three work-in-progress approaches that aim at elaborating such right requirements faster.

1   Introduction

In the last two decades, a multitude of requirements engineering (RE) approaches has emerged. Based on the commonly accepted observation that RE is indispensable for the success of a software development project, remarkable effort has therefore been spent on making requirements specifications more complete, more consistent, more correct, etc.

However, in recent years, the advent and wide acceptance of agile development approaches in software companies has, among other things, shown that industry is interested rather in "good enough" than in perfect requirements. In particular, requirements specifications and RE activities have taken a back seat, as they are no end in themselves, and often do not sufficiently satisfy the needs of developers anyway [1].

While these weaknesses do not imply that RE is not necessary for industry, modern RE approaches - at least in short-lived sectors such as the information systems domain - must stand out with high efficiency and pragmatism nevertheless.

In order to achieve this goal of higher efficiency, our idea is to improve the effectiveness of requirements approaches by constructively focusing on the right requirements. In this context, "right" means that only such requirements that are actually valuable for satisfying both external stakeholders and developers are engineered. Furthermore, "constructively" means that the entire requirements process is guided in a way that as little rework as possible is needed to achieve this set of "right" requirements.

In order to drive our research in this regard, three practical questions have been observed in RE practice, which we believe to be essential, but which have not been solved properly yet:

1.   *Which requirements are economically feasible?*

2. Which requirements are relevant for enabling early business success?
3. Which requirements are relevant for making development decisions?

In this paper, we present a research preview on how we are currently dealing with these practical questions. While we address the first question via constraints-aware elicitation, the second question is addressed via a model-based prioritization approach, and the third one via view-based specifications. The paper closes with an outline of how these approaches fit together and which benefit they have for a more efficient RE approach.

2 Research Preview

2.1 Constraints-aware Elicitation

Problem Elaboration. As a multitude of systems is nowadays built in a reuse-based manner instead of being developed from scratch [8], experience has shown that it is unrealistic that each system is actually able to satisfy all stakeholder requirements as initially stated. Rather, trade-offs between ideal requirements and rapid development must be made. However, in order to assess the economic feasibility of requirements in the context of given assets, knowledge about reuse capabilities and constraints is needed. Unfortunately, requirements engineers typically do not have such knowledge and thus need to involve development experts. Hence, as their assessment is mostly done offline, additional and late rework is often needed besides the "normal" rework that has to be spent due to changing stakeholder wishes anyway. In order to increase the efficiency of requirements elicitation, requirements engineers must therefore be enabled to make such assessments on their own directly during an elicitation session.

State of the Art. The most mature approaches for reuse are available in the area of software product lines (SPL), which have been proven to be the most strategic form of reuse. However, existing SPL RE approaches assume that the requirements that may occur during "development with reuse" can be anticipated explicitly during "development for reuse". However, as we have shown in our previous state-of-the-art survey [11], this assumption is often not fulfilled, which is why these approaches are not sufficient for solving the aforementioned problem. In particular, existing approaches do not explain how to extract and represent reuse capabilities and constraints systematically from a given reuse asset base in order to provide requirements engineers with corresponding knowledge.

Solution Idea. Our solution idea for solving this problem is to use a constraint-based rather than an enumerative approach for expressing the feasibility of requirements. Hence, instead of explicit listing of all requirements that are economically feasible, constraints are defined that restrict valid requirements declaratively. To make this happen, the idea is to provide a tailoring approach that prescribes a systematic method for extracting the characteristics of a given reuse asset base, and for reflecting them in a set of requirements elicitation instructions (see [11][12]).

Research Objectives. In order to realize this solution idea, the following research objectives must be achieved.

- *Alignment Model.* This model explains how RE processes are related with a given reuse asset base. By knowing these dependencies, we can define which requirements are economically feasible (see [13]).
- *Elicitation Instruction Template.* This template provides a generic structure as well as a set of predefined text blocks for representing best practices and important knowledge about a reuse asset base to requirements engineers in a suitable manner (see [14]).
- *Tool-supported Tailoring Method.* This method provides a clear sequence of activities to be carried out during "development for reuse" in order to derive a set of elicitation instructions according to the aforementioned template from a given reuse asset base.
- *Controlled Experiment.* This study evaluates whether requirements engineers using a set of elicitation instructions according to our approach are able to elicit requirements more effectively than when using state-of-the-art approaches.

Expected Benefits. The systematic extraction and explicit representation of reuse capabilities and constraints in an instruction document enables requirements engineers to be better aware of what is economically feasible and what is not. Hence, they are able to elicit and negotiate requirements more effectively. In particular, they can achieve a higher fit between requirements that are economically feasible by using the reuse assets, and those that are initially stated by the customer. Hence, less effort for costly re-implementations or late renegotiations is necessary, which leads to higher overall RE and development efficiency.

2.2 Model-based Prioritization

Problem Elaboration. The purpose of many software development projects is to build software to better support an enterprise's business processes in order to optimize business performance. Typically, such projects are characterized by high complexity – even in small and medium-sized enterprises, it is not uncommon to have several dozen business processes that need to be considered for optimization by possible system designs. In the area of RE and release planning, prioritization is an established strategy for assessing the best way to spend the available resources [10]. Decision makers in industry have difficulties in applying state-of-the-art prioritization techniques in such settings, leading to wasted time and effort spent on numerous (RE) activities of minor importance. In order to increase the efficiency of requirements elicitation, requirements engineers must be enabled to handle complexity by eliciting the most valuable requirements efficiently, i.e., in an optimal order.

State of the Art. In the literature, many prioritization techniques have been proposed, differing in terms of complexity, calculations, or their input and output, for example [8]. The selection and application of one special technique strongly depends on the application domain and the prioritization problem at hand [6]. However, despite the strengths of the techniques, most are designed to solve general requirements prioritization problems and thus are multi-purpose methods and do not support the complex requirements needed in business-process-driven development projects.

Solution Idea. The solution idea for tackling this prioritization problem is to provide a prioritization framework that takes into account the idiosyncrasies of such business-process-driven development projects. It shall support the requirements

engineer by providing him with information about how the particular requirements in such projects can be assessed (considering their dependencies and idiosyncrasies), which roles have to be involved in the prioritization process, and how the prioritization itself has to take place. This means that besides a concrete prioritization technique, further building blocks are integrated to build a comprehensive framework [7].

Research Objectives. In order to realize this solution idea, the following research objectives must be achieved.

- *Issue Model:* This model contains the typical issues (i.e., inherent elements that are either part of a system or part of the system's environment) relevant in business-process-driven RE, their relations among each other, and issue-specific information relevant for prioritization.
- *Value Model:* This model consists of the objective (measured) and subjective (assessed by stakeholders) criteria that are needed to rate requirements (concerning different issues) appropriately.
- *Role Model:* This model contains the different roles that are relevant for prioritizing the requirements concerning different issues.
- *Tool-supported Prioritization:* This method provides a way to conduct prioritization by using the information about issues, criteria, and roles provided in the models.
- *Controlled Experiment:* This study evaluates whether decision makers using the prioritization approach according to the solution idea are able to achieve an equally valuable product with less time and effort, or a more valuable product with the same time and effort.

Expected Benefits. Through the usage of issue-specific value criteria assessed by corresponding roles, requirements can be prioritized more appropriately. Requirements engineers are enabled to focus on the most valuable requirements. Hence, the overall RE efficiency increases, as time and effort are only spent on the elaboration of requirements that contribute to business success.

2.3   View-based Specification

Problem Elaboration. When creating requirements specifications (RS) within software development projects, different information needs have to be addressed. These information needs are strongly dependent on the particular role and task that development engineers (as the document consumers) have within the project. For example, an architect requires detailed information about quality and data requirements, while a user interface designer is rather interested in information regarding end user characteristics. However, today's RE approaches do not explicitly address these "role-specific" information needs. As a consequence, RS often contain more or even less information than actually required by a certain role to perform development tasks. Or the specified information is represented in an inappropriate form, such as lengthy text descriptions. All these problems negatively influence the efficient usage of the RS, as for example the analysis of the documents becomes time-consuming or even frustrating for the document consumers [5]. In the worst case, this problem could result in development engineers neglecting or ignoring the RS, which in turn could result in software implementations that fail to meet the requirements actually documented in the RS.

State of the Art. Existing approaches in the area of requirements specification provide more general answers to the content and representation of RS and basically propose "best practice" (e.g., [2] [3]). However, for the efficient development of novel information systems, these approaches might be too general, whereas specific RE approaches, e.g., for (self-) adaptive systems such as [4], might be too specific and neglect important information needs from the developers' viewpoint.

Solution Idea. To tackle the introduced problem, sound and empirically valid knowledge about particular information needs from the viewpoint of different development roles needs to be gained by means of suitable research activities. Such information needs can be expressed by certain artifact types (such as descriptions of stakeholders, interactions, quality attributes, or data) that should be specified in an RS to support engineers in performing their tasks adequately. Furthermore, knowledge needs to be gained about the respective level of detail and notation in which relevant artifact types should be specified. Based on this knowledge, suitable tool support can then be developed that, for instance, makes it possible to provide particular development engineers with RS that fit their particular demands by generating views [5] [7].

Research Objectives. In order to realize this solution idea, the following research objectives must be achieved.

- *Information Needs Analysis.* This analysis aims at identifying the information needs of different development roles in modern information systems development. For this analysis, suitable user studies have to be designed and conducted, e.g., via surveys, observations, document analysis, etc.

- *Information Needs Reference Model.* This model captures the knowledge about the role-specific information needs gained by the empirical studies conducted in the previous analysis activity.

- *Tool-supported Generation of Views.* This research objective aims to develop suitable tool support for generating views on RS. The vision is that the tool should support the demands of the various development engineers regarding their particular information needs. However, it might be difficult to develop a "one-fits-all" solution. Therefore, the tool might also provide features to adapt a personal view on an RS to specific (e.g., project-dependent) information needs.

- *Evaluation.* This research objective aims to investigate whether the expected benefits have been achieved or not. Suitable evaluation methods include, for instance, controlled experiments that compare "traditional" RS with "view-based" RS regarding variables like time required to create and analyze the RS or to find important information within the RS [5].

Expected Benefit. Consumers of view-based RS will be provided with all (and only) relevant information in an RS that supports them in performing their tasks. This leads to higher efficiency in RE and development, as the analysis of RS becomes faster (important information can be found easier). Furthermore, the creation of the RS itself could also benefit as the specification of the requirements could be tailored to the specific demands of the specification consumers.

3   Interplay and Conclusion

Each of the aforementioned solution ideas may provide significant benefits also in isolation. However, the idea of our research is to integrate them into a holistic approach for more efficient RE in the information systems domain, as the need for reuse-based [8] and lean approaches is very high here.

In Figure 1, the interplay between the solutions is therefore depicted. During the RE preparation phase, the requirements elicitation process is tailored. In this step, knowledge about the capabilities and constraints of the reuse asset base as well about the actual information needs in a certain domain is incorporated into elicitation instructions. During a concrete project, these instructions are then used to guide the process. At each stage of refinement, the requirements elicited until then are prioritized according to the criteria of the value model. This is done in order to spend further elicitation effort only on those requirements that promise the highest business value when implemented early. Thus, the elicitation can be streamlined, also taking existing constraints and capabilities into consideration continuously. The requirements that finally result from this process are then filtered and represented appropriately for the corresponding development roles based on their information needs. Hence, by applying this integrated concept, the elaboration of non-feasible, irrelevant, or useless requirements is avoided constructively, which finally increases the overall efficiency in RE.

We are aware that the tailoring required to address project or domain specifics will need additional effort that must be balanced with the intended efficiency improvements. So far, we only have first insights, but no final evidence on the cost/benefit ratio yet. However, as we expect that tailoring will not be needed for each project, large savings could be achieved within a certain domain.
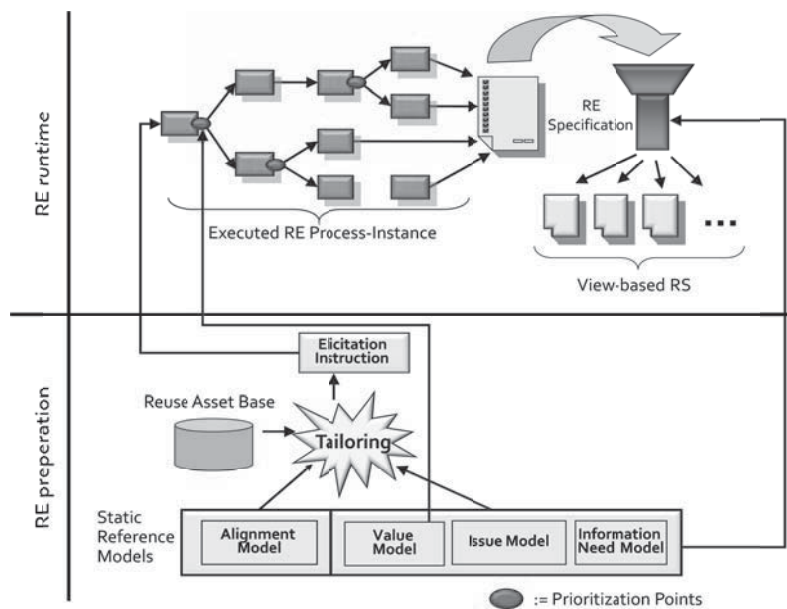
Figure 1. Interplay of Solution Ideas

Acknowledgment

References

1. Adam, S., Doerr, J., Eisenbarth, M.: Lessons learned from best practice-oriented process improvement in Requirements Engineering – A glance into current industrial RE application. In: Proc. of REET, IEEE, Atlanta (2009)
2. Robertson S., Robertson, J.: Mastering the Requirements Process. Addison-Wesley (2006)
3. IEEE Computer Society: Recommended practice for Software Requirements Specifications. Standard IEEE Std. 830-1998. IEEE, USA (1998)
4. Schmid, K., Eisenbarth, M., Grund, M.: From Requirements Engineering to Knowledge Engineering: Challenges in Adaptive Systems. In: Proc. of SOCCER'05, Paris (2005)
5. Gross, A.: Perspective-based Specification of Efficiently and Effectively Usable Requirements Documents. In: Proc. of Doctoral Symposium RE'10, Sydney (2010)
6. Salinesi C., Kornyshova, E., Choosing a Prioritization Method - Case of IS Security Improvement. In: Forum Proc.of CAiSE, pp.51 – 55. CEUR-WS.org, Luxembourg (2006)
7. Riegel, N., Adam, S., Gross, A.: Addressing requirements engineering challenges in the context of Emergent Systems. In: Proc. of RESS, pp. 6 – 9. IEEE, Trento (2011)
8. Sommerville, I., Lock, R., Storer, T., Dobson, J.: Deriving Information Requirements from Responsibility Models. In: Proc. of CAiSE, pp. 515 – 529. Springer, Amsterdam (2009)
9. Herrmann, A., Daneva, M.: Requirements Prioritization Based on Benefit and Cost Prediction: An Agenda for Future Research. In: Proc. of RE'08, IEEE, Barcelona (2009)
10. Riegel, N., Adam, S., Uenalan, O.: Integrating Prioritization into Business Process-driven Requirements Engineering. In: Proc. of REFSQ'10 Workshops, ICB, (2010)
11. Adam, S.: Improving SPL-based Information System Development Through Tailored Requirements Processes. In: Proc. of Doctoral Symposium RE'10, Sydney (2010)
12. Adam, S., Doerr, J., Ehresmann, M., Wenzel, P.: Incorporating SPL Knowledge into a Requirements Process for Information Systems – An Architecture-driven Tailoring Approach. In: Proc. of PLREQ Workshop at REfSQ'10, pp. 54 – 66. ICB, Essen (2010)
13. Adam, S.: Towards Faster Application Engineering through Better In-formed Elicitation – A Research Preview. In: Proc. of REFSQ'11 Workshops, pp. 19 – 24. ICB, Essen (2011)
14. Adam, S.: Providing SPL Knowledge to Requirements Engineers – A Template for Elicitation Instructions. In: Proc. of REFSQ'12. Springer, Essen (2012) (to appear)

GRCM: A Model for Global Requirements Change Management

Waqar Hussain, Tony Clear

Auckland University of Technology
{waqar.hussain,tclear}@aut.ac.nz
http://www.aut.ac.nz

Abstract. [**Context and motivation**] In the delivery driven context of contract software production, efficient and effective requirements change management (RCM) remains a challenge for global software development (GSD). [**Question/problem**] New RCM models need to be devised for GSD settings, to reduce confusion and improve the efficiency of managing requirements change and the resulting impacts. [**Principal ideas/results**] We present a model drawn from a case study which evaluated RCM practices in a GSD organization, with sites based in USA and Pakistan. [**Contribution**] We extend the observed practices by developing a theoretically informed process model to improve RCM efficiency and effectiveness by using a baseline requirements artifact and tool supported collaboration process.

Keywords: Global Software Development, Multi Site Requirements Change Management Model, Global Requirements Change Management Model, Requirements Engineering

1 Introduction

For software companies working in a global context, producing against tightly constrained software delivery contracts, requirements change management (RCM) is a critical task. Poorly handled change leads to reduced product and service quality, and unsatisfactory resourcing, technical and commercial outcomes. Recently there have been calls [1] for global software development (GSD) researchers to engage in practical partnerships, adapting existing methods and tools, rather than developing elegant theoretical models in isolation from practitioners.

This work investigates the RCM process as practiced in a GSD field setting and compares it with available RCM models (primarily suitable for single site development) from the literature [2–4, 11, 13]. We propose a global requirements change management (GRCM) model accommodating multi-site development extended from the activities, roles and artifacts identified in existing models for requirements change management [5].

2   Background

2.1   GSD and Requirements Management

GSD poses challenges for managing requirements change because distance (cultural, geographical, temporal and language) aggravates coordination and control problems, through its negative effects on communication [7]. Requirements management, one of the most collaboration-intensive activities in software development, presents significant difficulties when stakeholders are distributed [6].

Many partial solutions have been offered for the implementation of Requirements Engineering (RE) in a global environment but they lack process level detail [8]. GSD demands robust models, methods and processes that can efficiently and effectively execute GSD work [10]. This research responds to that need.

2.2   RCM Process Models

The RCM models found in the literature [2–4, 11, 13], are not designed for the GSD environment. Mapping these models to multi site development is difficult as they do not describe how the collaborative activity for managing change will be handled in a globally distributed project, and process level detail is missing. Yet practitioners are wrestling with these challenges on a daily basis.

A survey [5] was conducted that compared the various activities, roles and artifacts (ARA) in the existing process models of RCM. It was concluded that [12] gives the highest level of ARA coverage by a single model, (13 out of the total 34 elements found in the literature). It was further concluded that there were no standard models of RCM and lack of detail of the ARA involved reduced the value of these models for industrial practice.

Our proposed model is developed specifically for the GSD environment and is more comprehensive than the RCM models proposed in the literature (covering 24 of the 34 elements). It also prescribes the use of collaborative technology to more efficiently manage RCM activities across distributed sites. We believe this gives our model strength in reducing requirements management challenges arising from development projects conducted at a distance.

3   Research Process

We profile here the outcomes of an exploratory case study [14] aiming to enhance existing RCM models to better support GSD. The characteristics and context of the setting for this study are mapped below, followed by an elaboration of the data collected for the study.

3.1   The Case Study Settings

GSD Inc, the selected company for our case study, is a CMMI Level-II certified small to medium sized company with almost 100 employees. Two projects, *SDE*

(Project 1) and *DataDive 2.0* (Project 2) were observed during the case study. *SDE* is a web application development project for a leading publishing client organization in the USA. *DataDive 2.0* is a centralized web based application which provides a suite of tools for query and analysis. The GSD Inc Pakistan office undertakes development projects on a contract basis, to a client supplied specification, to meet the company's need for low cost solutions and additional expertise. The software development life-cycle is thus driven by up-front requirements, and negotiated pricing. In practice this results in a pragmatic version of *waterfall by feature* development, wherein changes with significant resourcing impacts result in renegotiation of pricing.

3.2 Data Collection and Analysis Methods

Data was collected for the two projects over a period of 8 months from August 2009 to April 2010 at the development site situated in Pakistan. A total of 36 change request forms were collected, 24 for project 1 and 12 for project 2. Our data analysis process, adapted from [14], investigated the change management process, related issues and the rationale for requirements change. Critical artifacts such as Change Request Forms (CRF), Software Requirements Specifications (SRS), email messages, status reports etc. were included for qualitative analysis of data. Semi structured interviews were conducted to support and validate this analysis. Key project members with at least three years experience in GSD, (the Change Moderator - *CM*, Quality Assurance Manager, Team Lead and Analyst) were interviewed.

4 The Proposed Global RCM Model

The company operated with a variable degree of adherence to CMMI prescribed RCM procedures. Issues identified with the existing RCM process in the study site were: insufficient impact analysis; limited sharing of information relating to rationale for changes; and poor recording of requirements change information. To address the inefficiencies introduced by these practices we propose a Global Requirements Change Management (GRCM) model for the GSD environment. The model draws upon frameworks from the literature, incorporating the typical change activities (namely *request, verify, implement, validate* and *update* [11]) of the normative RCM models [2–4] and extending the model presented by [13].

4.1 Description of The GRCM Model

The process model presented in Figure 1 uses the terms *Role* and *Site* to show the distribution of the work environment with multiple team members at multiple sites. In the inset at the top left corner the model shows (*Role1-Site1*) which means any *Role* (such as tester, developer, project manager) at any particular *Site* (Pakistan, US, India etc.) played by a stakeholder who can initiate change. Similarly (*Role2-Site2*) means any other key stakeholder role at a designated
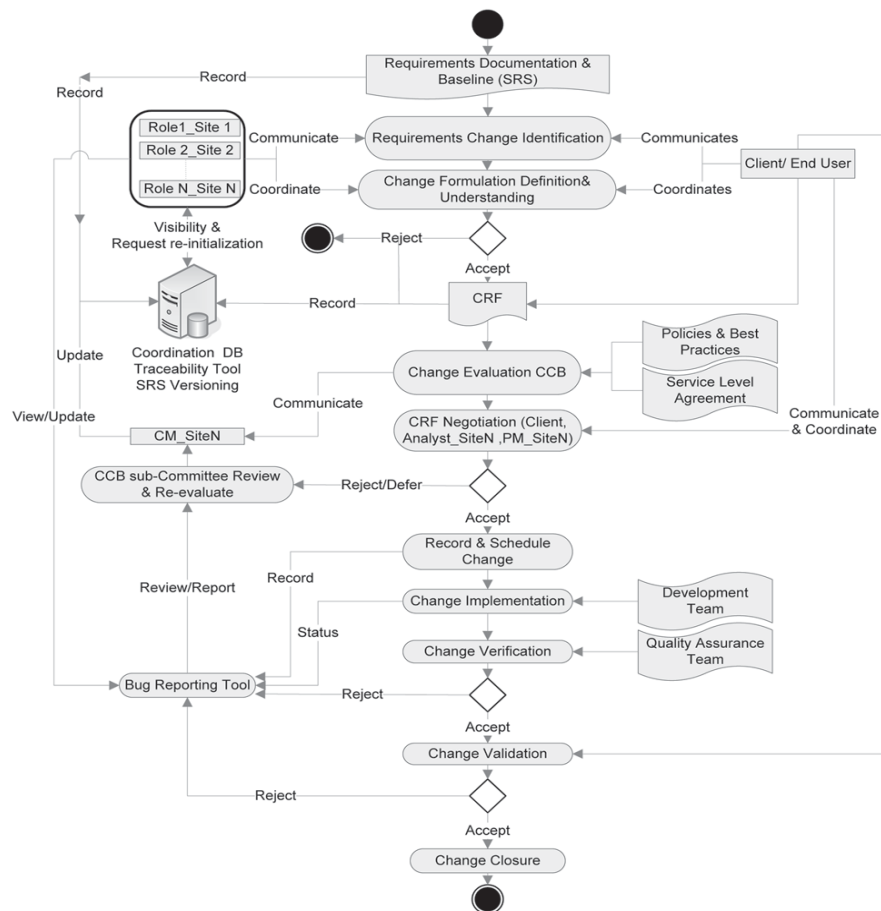
Fig. 1. Proposed Global Requirements Change Management Model (GRCM)

location (e.g. *Site2*). The model is extendable to include any number of teams, sites and stakeholders (*RoleN-SiteN*). In the proposed model only one client is shown for simplicity. However the model can equally reflect a number of clients at multiple locations, for example (*ClientN-LocationN*), and so on.

4.2 Operation of the GRCM Model

The model takes a baseline requirements document (in this case an *SRS*, but could equally include agile artifacts such as user stories) as an initial input into the process model. The baseline requirements artifact is linked with the coordination database to record and trace changes to the requirements. The *SRS* remains visible to all stakeholders across sites, once linked with this collaboration database, whereas specific design artifacts are visible to the local teams

only. When a change is identified and communicated by any stakeholder from a given site, it undergoes a process of *change formulation, understanding and definition.* This is a technology supported collaborative activity among distributed stakeholders. Upon its acceptance it moves to the formalisation stage, when a change request form (*CRF*) is filled out by the change initiator and submitted for formal review and evaluation by the change control board. The *CRF* is the key artifact circulated among the parties when considering a change. The requested change (whether accepted or rejected), is recorded in the online repository for future reference. The formally approved change request then enters the negotiation process. If the change is accepted for implementation it is recorded and scheduled using a tool which makes change data visible to all the stakeholders. After implementation by the development team it is verified and validated and then closed. If the change is rejected it goes to a subcommittee of the change control board for a review and re-evaluation process. The report is sent to the Change Moderator who then updates the coordination database and makes the status of the change available to all the stakeholders.

4.3 Application of the Proposed Model

The proposed GRCM model *Figure 1* may represent a variety of GSD contexts, and could be adapted to accommodate new roles identified in specific settings. The process model, with its support for collaboration through technology and shared artifacts, contributes to cross-site negotiations, awareness and visibility of changes. It provides a pragmatic balance between software production and control, thereby improving the efficiency of the RE process. While devised in a web application context, it is not limited to any organization or type of software project. Thus we believe it could be applied in a range of GSD settings.

4.4 Limitations of the Proposed Model

This GRCM model has been synthesized from theory and practice and has had some initial use within the case study site, to validate its effectiveness. The scope for testing and optimization of the process model still remains. The model applies primarily in support of RE activities and contract modification decisions, and thus has potential limitations in its applicability to the detail of later development phases. Yet within this study's constrained scope of pragmatic *waterfall by feature* development it provides a practicable approach. The model also lacks any prescription of the mediating technology that may be employed. Since many kinds of collaborative technologies (e.g. repositories, bug reporting tools etc.) can be used for GSD projects, we believe most organizations will tailor a technology set to suit their needs.

5 Conclusion and Future Work

Existing requirements change management models have not been specifically developed for the GSD environment. We report the findings from a case study that

investigated the change management process employed by a GSD organization. We identified several problems with their existing RCM process. We propose a resulting global requirements change management (GRCM) model, informed by our insights from theory and practice. The model incorporates the commonly adopted change activities (namely *request, verify, implement, validate* and *update* [11, 13]) of the normative RCM models [2–4]. The GRCM model augments these with a collection of activities, roles, and artifacts [13] from the literature. Currently the observations from its initial use at the case study site are encouraging and show signs of its efficiency and effectiveness in this industrial setting. The model now needs wider application in a variety of GSD project settings for a full assessment of its workability and scope of application.

References

1. Damian, D.: Requirements Engineering in Global Software Engineering: How far have we come? [Panel Session]. 6th IEEE International Conference on Global Software Engineering, IEEE, Helsinki, Finland (2011)
2. Olsen, N.C.: The software rush hour [software engineering]. Software, IEEE 10 (1993) 29-37
3. Mkrinen, M.: Application management requirements for embedded software. Technical Research Centre of Finland VTT Publications, Espoo (1996) 286
4. Ince, D.: An introduction to software quality assurance and its implementation. McGraw-Hill (1994)
5. Ramzan, S., Ikram, N.: Requirement Change Management Process Models: Activities, Artifacts and Roles. Multitopic Conference. IEEE Islamabad, Pakistan (2006) 219-223
6. Sengupta, B., Chandra, S., Sinha, V.: A research agenda for distributed software development. Proceedings of the 28th international conference on Software engineering. ACM, Shanghai, China (2006) 731-740
7. Carmel, E.,Agarwal, R.: Tactical Approaches for Alleviating Distance in Global Software Development. Software, IEEE 18 (2001), pp. 2229.
8. Lopez, A., Nicolas, J., Toval, A.: Risks and Safeguards for the Requirements Engineering Process in Global Software Development. 4th IEEE International Conference on Global Software Engineering, IEEE, Limerick, Ireland (2009)
9. Sangwan, R., Bass, M., Mullick, N., Paulish, D.: Global Software Development Handbook. Auerbach Publishers, New York, NY (2007)
10. Damian, D., Moitra, D.: Guest Editors' Introduction: Global Software Development: How Far Have We Come? Software, IEEE 23 (2006) 17-19
11. Niazi, M., Hickman, C., Ahmad, R., Ali Babar, M.: A Model for Requirements Change Management: Implementation of CMMI Level 2 Specific Practice. In: Jedlitschka, A., Salo, O. (eds.), Vol. 5089. Springer Berlin / Heidelberg (2008) 143-157
12. Leffingwell, D., Widrig, D.: Managing software requirements, A unified approach. Addison-Wesley, Boston (2000)
13. Imtiaz, S., Ikram, N., Imtiaz, S.: A process model for managing requirement change. Proceedings of the Fourth IASTED International Conference on Advances in Computer Science and Technology. ACTA Press, Langkawi, Malaysia (2008)
14. Briand, L.C., Basili, V.R., Kim, R.: A Change Analysis Process to Characterise Software Maintenance Projects. Presented at International Conference on Software Maintenance, Victoria, Canada (1994).

"Measurements of Effectiveness and Efficiency"-Driven Requirements Engineering and Test Plan Development

Oliver Furtmaier, Ren-Yi Lo

Siemens Corporation, Corporate Research and Technology, Princeton, NJ, USA
oliver.furtmaier@gmx.de
ren-yi.lo@siemens.com

Abstract. [**Context & Motivation**] In recent years, a lot of attention in requirements engineering (RE) has been given to the early understanding of problems. This is evident in the works on goal modeling [5], problem frames [2] and problem oriented software engineering [1], respectively. [**Question/problem**] The objective is to detect and resolve conflicts earlier in the development and create a more consistent rationale for the high level requirements in order to make early design decisions possible, traceable to the problems and transparent to all stakeholders. [**Principal ideas/results**] This should be done by involving the stakeholders, requirements engineers and testers in the derivation and evaluation of testable, problem-oriented selection criteria from stakeholder problems, which are referred to as measurements of effectiveness and efficiency. These criteria set the direction for the development of a solution and measure if any solution has satisfyingly solved the problems. Hence they drive requirements engineering as well as testing. The application to a fictitious camera specification has bridged crucial gaps in the business rationale. [**Contribution**] This process has been further developed from the measurements of effectiveness approach by Noel Sproles [3, 4] and enhanced towards efficiency. Furthermore, areas of future investigations have been identified for this research preview.

References

1. Hall, J. G., Rapanotti, L., & Jackson, M. A. (2007). Problem Oriented Software Engineering: A design-theoretic framework for software engineering. *Fifth IEEE International Conference on Software Engineering and Formal Methods*, (pp. 15-24).
2. Jackson, M. A. (2001). Problem Frames: Analyzing and Structuring Software Development Problems. London: Pearson Education.
3. Sproles, N. (2000). Coming to Grips with Measures of Effectiveness. *Systems Engineering , 3*, 50-58.
4. Sproles, N. (2002). Formulating Measures of Effectiveness. *Systems Engineering , 5*, 253-263.
5. van Lamsweerde, A. (2001). Goal-Oriented Requirements Engineering: A Guided Tour. *5th IEEE International Symposium on Requirements Engineering*, (pp. 249-263). Toronto.

4 Creativity in Requirements Engineering (CreaRE)

Editors

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

Joerg Doerr
Fraunhofer Institut IESE, Germany, joerg.doerr@iese.fraunhofer.de

Andrea Herrmann
Infoman AG, Germany, andrea.herrmann@infoman.de

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

Workshop Programme

CreaRE 2012
2nd Workshop on Creativity in Requirements Engineering

Maya Daneva[1], Joerg Doerr[2], Andrea Herrmann[3], Kurt Schneider[4]

[1] University of Twente, Netherlands, m.daneva@utwente.nl
[2] Fraunhofer Institut IESE, Germany, Joerg.Doerr@iese.fraunhofer.de
[3] Infoman AG, Germany, andrea.herrmann@infoman.de
[4] Leibniz Universität Hannover, Germany, kurt.schneider@inf.uni-hannover.de

1 Technical Program

The CreaRE workshop took place as a half-day workshop on the 19th March 2012 in Essen (Germany). The agenda included four paper presentations, a keynote talk and an improvisation theatre session:

- *Daniel Berry* (keynote): Are Creativity, HCI, and Emotions Parts of RE? — Are Requirements Invented or Discovered?
- *Alessia Knauss (Olesia Brill), Eric Knauss, Daniela Damian:* Towards Supporting End-User Creativity with Social Media and Multimedia
- *Li Zhu, Thomas Herrmann:* Design Now! — Elaborating Requirements in Situated Action
- *Deepti Savio, P.C. Anitha:* 'Pictionades': Enhancing Stakeholders' Awareness about Issues in Requirements Communication
- *Sylviane Levy, Fernando Gamboa:* Requirements Analysis for Multimedia Interactive Informative Systems: a Metamodelling Approach
- *Anne Hoffmann, Martin Mahaux:* Research Preview: Using Improvisational Theatre to Invent and Represent Scenarios for Designing Innovative Systems

2 Introduction

Requirements Engineering (RE) not only demands a systematic approach for eliciting, operationalizing, and documenting requirements and for solving their conflicts, but RE 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 2012 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 served as a forum for the exchange of experiences and research results. It also aimed at raising awareness in the RE community for the importance of creativity and creativity techniques. Last, the workshop reached out and made a first step towards linking the RE community to other communities to which creativity is essential.

We invite readers to review the CreaRE 2012 web site for further information: http://www.se.uni-hannover.de/events/creare-2012/index.php/Introduction

3 Targeted Audience

CreaRE's long term vision is to bring together practitioners and researchers from both the RE community and other related communities, for example, creative design, psychology, design thinking, to debate on how to leverage creativity approaches for the purpose of better RE. The workshop organizers are committed to provide opportunities for practitioners to learn about pragmatic ways for incorporating creativity techniques into RE processes. To researchers, the workshop provides a forum to discuss relevant and under-researched RE phenomena where creativity is of central importance.

4 Program Commitee

We thank our program committee members for their support:

D. Berry (University of Waterloo, Canada
D. Callele (University of Saskatoon, Canada)
A. Hoffmann (Siemens, Germany)
D. Kerkow (Fraunhofer Institut IESE, Germany)
R. Ocker (Penn State University, USA)
K. Schmid, University of Hildesheim, Germany)
I. van de Weer (University of Utrecht, Netherlands)
R. Wieringa (University of Twente, Netherlands)
K. Zachos (City University London, UK)

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.

5 Keynote Presentation: "Are Creativity, HCI, and Emotions Parts of RE? — Are Requirements Invented or Discovered?", by Daniel Berry

This keynote talk offered a variety of perspectives on the question of whether creativity is part of RE at all. The talk suggested that creativity is indeed part of RE, if requirements are something that is to be invented. Berry defined creativity as the generation of innovative, unexpected solutions to complex, non-trivial problems, or to ill-formed, wicked problems. Dan Berry — and many RE researchers who consider RE as a socially constructed activity — think that creativity is an integral part of RE. Examples from Berry's own research were presented in support of this viewpoint. Berry also shared personal evidence suggesting that there are different opinions on whether the topics of inventing requirements, reasoning about emotional requirements, or using personas in RE is part of RE and whether papers on these topics should be published in RE outlets or elsewhere. Because whether creativity is a part of RE is debated in the RE community, Berry invited the RE community to work towards increasing the awareness of the role that creativity and creativity techniques can play in RE. He emphasized that workshops on creativity should become part of any RE event. Furthermore, Berry offered his reflections on the history of research about creativity in RE. One of the reasons why RE needs creativity is that RE is a wicked problem for any non-trivial software-intensive system. Any wicked problem demands abandoning old ideas and finding innovative ways to solve problems. Creativity can even happen when someone fails to follow conventions. Errors can lead to new ideas. Creativity not only produces large numbers of requirement ideas but also provides the methods to cope with this avalanche of ideas. Therefore, creativity must be fostered instead of controlled or even banned. Berry concluded that requirements are both invented as well as discovered.

Towards Supporting End-User Creativity with Social Media and Multimedia

Alessia Knauss, Eric Knauss, Daniela Damian

SEGAL, Dept of Computer Science, University of Victoria, Canada
`{alessiak,erickn,danielad}@cs.uvic.ca`

Abstract. When improving existing software systems, requirements engineers have to capture stakeholder needs. These needs have to be transformed into improvements of the system. Creative processes accompany this task. Especially when improving large systems with many heterogeneous stakeholders, it is difficult to consider all stakeholders. End-users of the system can be a valuable source of creativity in discovering requirements, currently not sufficiently supported in conventional requirements engineering methods. Today, these end-users are adept in using new techniques (e.g. multimedia, and social media). This allows using these techniques to establish a community of practice, facilitate creativity among end-users, and leverage this source of creativity in requirements engineering. In this paper we describe our vision on how to support end-users by leveraging novel modes of interaction such as social media and multimedia. We propose a number of research questions grounded in related work in the areas of creativity, social media and multimedia.

Keywords: Multimedia; End-User Participation; User-Centered Requirements Engineering; Social Media; Seeding

1   Introduction

According to Sawyer and Kotonya [1] systems are often unsatisfactory because requirements for one group of stakeholders have been stressed at the expense of others. This problem is even more complex, because modern software systems are increasingly large-scale systems with many different groups of stakeholders. One of the main challenges of requirements engineering for these types of systems is to identify the requirements of *all* stakeholder groups. In this position paper we discuss, how to involve a special stakeholder group in requirements engineering – the end-users – and their creativity in requirements engineering. Plucker [2] defined creativity as "*the interplay between ability and process by which an individual or group produces an outcome or product that is both novel and useful as defined within some social context*". Previous research on creativity showed promising support for requirements engineering (e.g. [3–6]). Yet, it remains to be investigated how to include the creativity of a representative set of end-users.

Recently, new approaches have been proposed that leverage multimedia [3, 7, 8], social media [9, 10], and underlying social networks [11, 12] for requirements engineering. Maalej and Pagano [9] propose a process that enables engineering teams to systematically gather and exploit user feedback in the software lifecycle. For this,

they integrate social media into software systems and the engineering infrastructure. They also integrate observations of user interactions while using the software and proactively collect in situ feedback. UserVoice[1] is one example of a social media tool that allows users to give feedback as support for requirements elicitation. We take the appearance of such tools as an indicator that a market exists for the kind of topics described in this position paper. Lim and Finkelstein [11] take these concepts one step further and offer empirical results. They propose to use StakeRare, a social network for requirements elicitation and prioritization that leverages snowball effects. Stakeholders were found to be cooperative (79% responses) in using StakeRare. Compared to conventional methods (e.g. workshops or interviews), stakeholders spend less time for requirements elicitation when using this method and preferred the new method over the conventional method.

The fact that stakeholders prefer social media suggests that this might be a suitable technique to support end-users' participation in requirements engineering and an opportunity for us researchers to leverage it. In this paper we propose to investigate if social media can support end-user creativity in requirements engineering. More precisely, we are interested in investigating seeding of social media for requirements engineering. That is, what kind of input (e.g. multimedia) should be present in social media to support their users' creativity?

2   Support for End-User Creativity

Nguyen and Cybulski [13] reflect upon the changing role of users in requirements elicitation. They argue that users are no longer *passive sources of requirements information*. Further, the emergence of new social media (such as YouTube, Wikis and Blogs) leads to a new type of users, the *naïve analysts*. These users are comfortable with creating contents. A success factor for requirements elicitation with these naïve analysts is the ability to closely collaborate and to be part of a wider learning community, which is creative and imaginative. Zarvic et al. [14] design the collection of requirements as a game. This encourages stakeholders to participate and supports creativity and the identification of hidden requirements. Similar effects might be visible with end-users who participate in requirements engineering supported by social media: They might feel less pressure and enjoy the opportunity to articulate their needs. This would have a positive impact on their creativity and on the effectiveness of requirements elicitation activities. Maiden et al. [4] give a mapping between software development processes and stages of an established creativity method (the CPS method). Based on this mapping, they identify opportunities to support requirements engineering with creativity. End-user participation is beneficial during *objective finding* (i.e. goal modeling), *fact finding* (i.e. requirements elicitation), *problem finding* (i.e. goal modeling), and *idea finding* (i.e. requirements refining and decomposition).

When this task is supported by social media, it resembles an evolving knowledge base. Fischer [15] suggests that such evolving knowledge systems need to be initialized with relevant content – the seeding. Therefore there are important research questions that arise in the study of end-user participation in requirements engineering

---

[1] http://uservoice.com/

and which relate to how social media, multimedia and seeding can be used to facilitate end-user creativity. We explore these questions in detail in the remaining sections.

2.1 Social Media: Infrastructure for End-User Creativity in RE

Shneiderman [23] argues that creativity works best when people interact. In his creativity framework, he proposes an explicit step where the person to be creative consults with peers. Social media is well suited for this task, because supporting interaction between users is their basic idea.

Creativity is a social process [16]. A good group formation can have a high impact on the groups' creativity. For requirements engineers it is hard to figure out which end-user groups should discuss specific requirements. Coordinating all constellations of discussions (as e.g. in [5]) is significant effort. In contrast, one of the key features of social media is bringing together people with similar interests. In requirements engineering this offers a chance for stakeholder groups to emerge based on their common domain expertise. We propose to use this for supporting creativity in requirements engineering and integrate support for creativity techniques in social media (e.g. based on the works of Schmid et al. [17, 18]). Social media can support the creativity process in spite of spatial distance. Social media supports end-users' participation without pressure and in an asynchronous manner, thus making it easier for end-users to get involved. The question is, whether this work is creative:

- *Research Question 1:* How can social media be leveraged effectively to stimulate creativity in requirements engineering?

First results reported in related work are promising: Lohmann et al. [10] use a wiki as social media that allows stakeholders to submit and discuss their requirements. They use this technique for projects with a defined scope and set of stakeholders. They report good results from letting stakeholders discuss and rate requirements in the SoftWiki. Solis and Ali [24] extend their Spatial Hypertext Wiki with creativity techniques. Singer et al. [19] take such concepts further by investigating, how the power of innovation in social networks can be leveraged. They argue that this is an important asset for identifying innovative features for increasing the competitiveness of systems.

It thus becomes important that research investigates systematically whether social media has a positive influence on creativity, if such social media tools would bring together people with conflicting or without common interests and how to add support for end-user creativity (e.g. seeding of content, for example multimedia).

2.2 Multimedia: Stimulation of Creativity in Requirements Engineering

Maiden et al. [3, 6] report that using multimedia during scenario walkthroughs leads to better results (i.e. more requirements). Furthermore, multimedia allows to capture context of a missing requirement and to express how the system should work in this context [7, 8]. End-users might like to use multimedia, because it makes it easy to capture context. A typical example is including a screenshot and referring to it when describing future needs. In this way, multimedia enables end-users to express themselves at low cost [7]. With the high availability of smartphones with good

cameras and the ability to access content in the internet, mobile devices are becoming another valuable source for multimedia in situ requirements [20, 21]. Based on these works, the following research question arises in the context of this paper:

- *Research Question 2:* How can multimedia be leveraged effectively to stimulate creativity in requirements engineering?

Research should systematically investigate the impact of multimedia requirements on creativity. We assume that a multidisciplinary approach including work from psychology and cognitive science is most promising.

2.3 Seeding: Preparing a Fertile Information Base for Creativity

Fischer [15] argues that complex systems need to evolve. Therefore, he uses the term *knowledge construction* in contrast to *knowledge acquisition*. That is, knowledge is only built during the lifetime of the system, instead of requiring domain experts to articulate all requirements a priori. Further, he shows that a solid information base is beneficial for this knowledge evolution – the seed. This gives users something to react – a prerequisite for capturing tacit knowledge. Experts can be made aware about their tacit knowledge when a breakdown occurs while they apply this knowledge. We can consider the continuous gathering of requirements from end-users in social-media as knowledge construction, i.e. the construction of knowledge how the system should be. It remains an open question how to do the seeding for this special type of knowledge construction and what kind of input is appropriate. Sources for input can be an initial set of ideas for improvement from the requirements engineers or a number of relevant bug reports. A promising alternative is using data from in situ feedback tools (e.g. [9, 20, 21]). These tools gather objectives, facts, problems, and ideas during usage of the system that should be improved or exchanged. The feedback can also include multimedia content and can be used for seeding at a low cost. Such in situ feedback can provoke breakdowns with end-users that might have experienced similar situations. Especially, when enriched with multimedia content, in situ feedback allows end-users to put themselves in the position of the sender.

If confronted with a blank screen, end-users might be discouraged to invent new desired objectives or ideas and creativity disappears. Therefore, we assume that good seeding has high impact on the end-users' creativity.

- *Research Question 3:* How can creativity in requirements engineering be stimulated by seeding of initial content in social media?

Especially when we think about multimedia as a seed for creativity in social media.

3 Proposed Research Method

We propose to investigate these research questions through case study research, as this allows observing and analyzing phenomena in a realistic context. Supporting end-user creativity in requirements engineering can be regarded as a process improvement endeavor. Therefore, the Goal-Question-Metric [22] paradigm could offer a suitable research method for such case studies.

One of the main challenges we currently see is finding a suitable set of metrics to measure creativity. Based on Plucker and Beghetto [2], we plan on measuring the novelty and usefulness of contributions based on questionnaires (cf. Section 1).

First we plan to investigate if seeding social media with multimedia content leads to more (creative) end-user requirements compared to seeding with text-based content. For this evaluation purpose existing social media (for example StakeRare or Facebook combined with YouTube) can be used.

4   Conclusion and Outlook

Using multimedia and social media in requirements engineering is a promising and emerging field. This shows in a number of related works that recently appeared. In contrast to related work, we focus on creativity of end-users. We propose to use in-situ feedback as a seed to create a fertile information base that allows creativity. Especially, when this in-situ feedback contains multimedia content, we expect a positive effect on creativity. We focus on end-users, because they are the best domain experts concerning the evolution of software systems. Especially in systems with a large user base, social media promises to reach a better sample of end-users than conventional requirements engineering methods. In addition, social media can support the social nature of creativity, even in the face of spatial distribution.

In this paper, we highlighted key concepts and motivated a number of research questions grounded in the current state of research in creativity in requirements engineering and the impact of seeding multimedia in social media.

5   References

1. Sawyer, P., Kotonya, G.: Software Requirements. Neurosurgery Clinics Of North America. pp. 179-86. Microsoft Press (2001).
2. Plucker, J.A., Beghetto, R.A.: Why Creativity Is Domain General, Why It Looks Domain Specific, and Why the Distinction Does Not Matter. Creativity: From Potential to Realization. 153-167 (2004).
3. Zachos, K., Maiden, N., Tosar, A.: Rich-Media Scenarios for Discovering Requirements. IEEE Software. 22, 89-97 (2005).
4. Maiden, N., Jones, S., Karlsen, K., Neill, R., Zachos, K., Milne, A.: Requirements Engineering as Creative Problem Solving: A Research Agenda for Idea Finding. RE'10. pp. 57-66 (2010).
5. Mich, L., Anesi, C., Berry, D.M.: Requirements Engineering and Creativity: An Innovative Approach Based on a Model of the Pragmatics of Communication. In: Regnell, B., Kamsties, E., and Gervasi, V. (eds.) REFSQ'04. pp. 129-144. , Riga, Latvia (2004).
6. Karlsen, I.K., Maiden, N., Kerne, A.: Inventing Requirements with Creativity Support Tools. In: Glinz, M. and Heymans, P. (eds.) REFSQ'09. pp. 162-174. Springer, Amsterdam, The Netherlands (2009).
7. Brill, O., Schneider, K., Knauss, E.: Videos vs. Use Cases: Can Videos Capture More Requirements Under Time Pressure? In: Wieringa, R. and Persson, A. (eds.) REFSQ'10. pp. 30-44. Springer, Essen, Germany (2010).
8. Creighton, O., Ott, M., Bruegge, B.: Software Cinema-Video-based Requirements Engineering. RE'06. pp. 106-115. IEEE Computer Society, Minneapolis, Minnesota, USA (2006).

9. Maalej, W., Pagano, D.: On the Socialness of Software. International Conference on Social Computing and its Applications. IEEE, Sydney, Australia (2011).
10. Lohmann, S., Heim, P., Auer, S., Dietzold, S., Riechert, T.: Semantifying Requirements Engineering: The SoftWiki Approach. I-SEMANTICS'08. pp. 182-185. Graz, Austria (2008).
11. Lim, S., Finkelstein, A.: StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation. IEEE Transactions on Software Engineering. pp. 1-32 (2011).
12. Lim, S.L., Damian, D., Finkelstein, A.: StakeSource2.0: Using Social Networks of Stakeholders to Identify and Prioritise Requirements. ICSE'11. pp. 1022-1024. ACM Press, Waikiki, Honolulu (2011).
13. Nguyen, L., Cybulski, J.: Into the Future: Inspiring and Stimulating Users' Creativity. Pacific Asia Conference on Information Systems. pp. 123-135. Suzhou, China (2008).
14. Zarvić, N., Duin, H., Seifert, M., Thoben, K.-D., Bierwolf, R.: Collecting end user requirements playfully. ICE Conference'09 (2009).
15. Fischer, G.: Seeding, Evolutionary Growth and Reseeding: Constructing, Capturing and Evolving Knowledge in Domain-Oriented Design Environments. Automated Software Engineering. 5, pp. 447-464 (1998).
16. Fischer, G.: Social creativity: turning barriers into opportunities for collaborative design. Lifelong Learning. 1, pp. 152-161 (2004).
17. El-sharkawy, S., Schmid, K.: A Heuristic Approach for Supporting Product Innovation in Requirements Engineering: A Controlled Experiment. In: Berry, D. and Franch, X. (eds.) REFSQ'11. pp. 78-93. Springer, Essen, Germany (2011).
18. Grube, P.P., Schmid, K.: Selecting Creativity Techniques for Innovative Requirements Engineering. 3rd International Workshop on Multimedia and Enjoyable Requirements Engineering (MERE '08). pp. 32-36. , Barcelona, Catalunya (2008).
19. Singer, L., Seyff, N., Fricker, S.A.: Online social networks as a catalyst for software and IT innovation. 4th International Workshop on Social Software Engineering. pp. 1-5. ACM, New York, NY, USA (2011).
20. Schneider, K., Meyer, S., Peters, M., Schliephacke, F., Mörschbach, J., Aguirre, L.: Feedback in Context: Supporting the Evolution of IT-Ecosystems. PROFES. pp. 191-205 (2010).
21. Seyff, N., Graf, F., Maiden, N.: End-user requirements blogging with iRequire. ICSE'10. pp. 285 (2010).
22. van Solingen, R., Berghout, E.: The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development. McGraw-Hill Publishing Company (1999).

Design Now! – Elaborating Requirements in Situated Action

Li Zhu, Thomas Herrmann

Dipartimento di Informatica e Comunicazione, Università degli Studi di Milano
Via Comelico 39/41 20139 Milano, Italy
`zhu@dico.unimi.it`
Information and Technology Management, Ruhr-University of Bochum,
Universitaetsstr. 150, 44780 Bochum, Germany
`thomas.herrmann@rub.de`

Abstract: This paper presents an empirical study on how to elaborate ideas for requirements with a creativity oriented meta-design environment, MikiWiki (Zhu 2011). MikiWiki was applied for the collaborative interface design of the *Creativity Barometer* (Herrmann et al. 2011) in a co-located meeting context. Through five collaborative design sessions, we aimed to observe how meta-design principles support collaborative creativity in practice. This empirical study is valuable in advancing our understanding of how meta-design fosters creativity and supports identifying requirements of various stakeholders. Our findings indicate that a meta-design approach not only enables requirements engineering at use time but also enhances different levels of creativity: 1) opportunistic programming as bricologe (Lévi-Strauss 1968) at the meta-design level, in that meta-designers constantly evolved the MikiWiki design environment opportunistically to cope with emergent socio-technical issues without needing to change server-side code; and 2) creativity-in-use at the design and use level, in that designers and users invent their own ways to use MikiWiki which are not envisioned by meta-designers. In addition, a more visual-based approach is appropriate to involve different design communities and enhance creativity.

Keywords: Design Now, meta-design, collaborative design, creativity, MikiWiki, requirements

1   Introduction

Future uses and problems cannot be completely anticipated at the software design time, thus requiring software environments that can be evolved at use time (Bourguin et al. 2001). The co-evolution of systems and users' social practices challenges requirements engineering (RE).

Since it is unrealistic to come up with fully described requirements for yet unknown problems and a continuously changing context, it is necessary to extend the RE-process in use time, providing possibilities to accommodate emergent new requirements.

Meta-design is an approach that strives to create social conditions and design processes for broad participation in design activities at both design time and use time, rather than anticipating all design requirements at design time (Fischer et al. 2004). The characteristics of meta-design are described in detail in (Fischer and Herrmann

2011). With respect to the presented case study and the support of creative RE it is crucial that:

1) Meta-design with respect to software engineering does not deliver fixed solutions but a set of tools which enables domain experts and their supporters to produce iteratively improving applications, in accordance with their evolving needs.

2) Meta-design implies design-in-use: it helps to continuously adapt design environments. The adaptation can be closely coupled with the usage of the design environment itself.

3) Meta-design provides a communication space for artifacts based, participatory design where end users are empowered to be designers.

RE therefore in this paper is twofold. Firstly, using a meta-design system to rapidly collect and externalize expectations for a software system; these expectations are mainly visualized (via short notes, symbols, sketches etc.) and can later on be systematically described with text, tables etc.. Secondly, the socio-technical challenges (Herrmann 2009) that become obvious during design sessions can be used to generate software requirements for improving the meta-design environment itself in the context of use time. However, the relationship between meta-design and RE has not been intensively explored, though a hint can be found in (Peffers et al. 2007).

The contributions of this paper are the following:

1) It demonstrates that the feasibility of evolving RE through a meta-design approach. We use *"Design Now"* to demonstrate our attempt. This refers to meta-design (Fischer et al. 2004) by emphasizing the immediacy and situatedness of bringing the usage perspective into design and the design perspective into usage. This immediacy is a decisive prerequisite for the involvement and creativity of all the participating stakeholders.

2) Moreover, RE requirements are typically represented via use cases and textually described. In contrast, the approach we explored does not aim at developing textual descriptions of requirements, but rather relies on more indirect descriptions via symbols, sketches, short notes, images and so on. Our findings demonstrate that a visual-based approach is appropriate and effective in involving different design communities and in supporting them to create visions of a future software system, as well as in imagining its central characteristics and implications, in particular some soft and hard to capture concepts, e.g. emotions.

Section 3 introduces MikiWiki (Zhu 2011), a web-based meta-design environment with which we conducted our case study consisting of five co-located meeting sessions. Section 4 explains the methodology of our case study and related information about design sessions. Section 5 describes some findings from the case study and a brief discussion is introduced in section 6.

2   Background

Suchman emphasizes situatedness of design action, in that the users' work and behavior is contingent on a complex world of objectives, artifacts and other actors located in space and time (Suchman 1985). Situated action is how actors act in a situation. It stresses the knowledge ability of actors and how they use commonsense practices to produce, analyze and make sense of one another's actions and their situated context (Doerry 1995).

Since the circumstances of users' actions are never fully anticipated and are continuously changing, it is necessary to design systems to accommodate the unforeseeable contingences of situated actions (Suchman 1985).

Situated design (Pfeifer and Rademakers 1991; Müller and Pfeifer 1997) is a design methodology for software engineering. It capitalizes the notion of the human as a situated agent. It implies that initial plans of actions are quickly abandoned once the work of design is underway. The general steps are: 1) Developing a vision of where you want to go; 2) Analysis of the complete working situation and initialization of the process; 3) Designing the initial system; 4) Introduction of the system into the working environment; and 5) Evaluation, taking into account the new working environment and generating ideas about new system (Müller and Pfeifer 1997).

However, the characteristics of new software do hardly become automatically apparent by just considering the situation in which it will be needed. A successful solution needs to be based on creativity. The creative process should take place as close as possible to the situation of software usage and design decisions. By situated creativity we mean that new ideas are immediately visualized in the design context so that they can talk back to their creator and that they are perceivable to other participants who also can contribute their feedback. To make such immediate feedback possible, the ideas can only be roughly outlined. They are refined step by step within a series of trail-and-error actions, which makes it similar to bricolage (Lévi-Strauss 1968) – that is a preliminary solution is drafted with simple means to understand whether it is sufficient or not. The character of preliminarity is constitutive for creativity and bricolage. Further, the focus on visual externalizations implies that the included users or stakeholders are focused on how the functionality of the system is mirrored by the user interface of the system.

3   MikiWiki

MikiWiki is chosen as it best serves our purposes in this paper. It provides a concrete meta-design environment, in which requirements or expectations can be visualized as well as textualized. It directly supports creative and collaborative drafting of the features of a software system.

MikiWiki is a structured programmable wiki to concretize the main meta-design characteristics. Beyond providing tools for text content production as in traditional wikis, MikiWiki allows all the stakeholders to collaborate in practice design and to continuously evolve the whole wiki system.

For the purpose of this paper, we only briefly introduce one distinctive feature of MikiWiki, "nuggets". In analogy with Lego construction kits, providing simple parts with which the user can create complex artifacts (Resnick et al. 2005), nuggets are the building blocks of MikiWiki shared between stakeholders.

To support collaborative RE, nuggets address collaborative design from different aspects. As an example, fig. 1 demonstrates participants designing a mobile interface with various nuggets, e.g. PostIt *note*, different *toolbox*, *canvas* and *trash* nuggets, etc. Participants can utilize the *sync-imagenote* nugget to create moodboards, or the *doodle* nugget to visualize abstract concepts (fig. 2). This not only helps them to express emotional attitudes but also to understand their expectations towards the system. A decisive characteristic of nuggets is that the representation of ideas, which can be created with different nuggets, can be interrelated to each other. Therefore

nuggets can intertwine the various perspectives of different stakeholders and they can bridge various phases of the design.

With these building blocks, users can use, mix and modify them, adding new behaviors or creating new ones. Nuggets therefore become a medium to facilitate introducing emergent requirements for MikiWiki at use time as well as collecting and prototyping requirements of design projects.

4   Case study

The design study was done in the Information and Technology Management Group at the Ruhr-University of Bochum, Germany. Meta-designers, designers and users were tasked to collaboratively redesign for a mobile version of a micro-survey tool, the *Creativity Barometer* [2], as part of an ongoing design project.

4.1   Context of the Case Study and Goals

The purpose of the *Creativity Barometer* is to conduct surveys to continuously understand and assess the climate of a company's creativity support. The *Creativity Barometer* allows companies to periodically repeat surveys and get instant feedback continuously. After a pre-specified time period (e.g. eight months), the company can summarize the feedback and plan interventions to improve the creativity climate. Since continuous surveying can disturb the employees the idea is to support them to give their answers as "en passant" as possible, e.g. with smart phones. The *Creativity Barometer* was first evaluated with a desktop-based web browser. It was successfully used in 4 companies where for instance 99 employees produced 2673 answers in September 2011. Therefore, transferring the desktop-oriented browser-version to smart phones appeared reasonable. However, the main concern we had was that users would stick with their impression of the already known solution when being asked for their expectations towards a smart phone version. Therefore we have considered the context of this design task as a reasonable case where creativity techniques should be applied. This design task – drafting the appropriate characteristics of the smart phone solution – has been chosen to evaluate meta-design in MikiWiki and to understand how far MikiWiki could contribute to the discovery of requirements.

Our design study questions are:
1)   Whether MikiWiki supports a transition between design for use and design in use, thus making RE an iterative and ongoing process;
2)   Whether lightweight tools provided by MikiWiki allow participants with different background and different roles to articulate and share their ideas and needs;
3)   How far MikiWiki supports creativity of meta-designers, designers and users.

4.2   Environment Setting

The design study was conducted in the modlab in the Department of Information and Technology Management, Institute of Applied Work Science at the University of Bochum. Five collaborative design sessions supported by MikiWiki were conducted and evaluated in a co-located collaboration context. A large, high-resolution interactive wall (4,80m x 1,20m; 4320x1050 pixels) seamlessly integrates three rear projection boards (see fig. 1). The touch screen displayed the MikiWiki mockup environment. Data can be entered and manipulated directly on the screen or via iPads

which are connected via WLAN (Herrmann 2010). Most important, the developing content of the large screen as well as the participants' activities can be completely recorded with the modlab. The recordings support a systematical analysis for detecting requirements afterwards while the session itself can be run in an associative, non-linear mode.

4.3 Methodology

This design study follows an action research approach (Avison et al. 1999). Action research is a framework for information system research that includes the expansion of social scientific knowledge as well as practical problem solving in social settings (Avison et al. 1999). Action research is an iterative process involving researchers and participants collaborating together on a particular cycle of activities, e.g. problem diagnosis, reflective learning. The essence of action research is a two-stage process (Blum 1955): 1) The diagnostic stage, in which the usage of the environment by the participants was observed and they were afterwards interviewed; and 2) the therapeutic stage, in which videos and the recorded interviews were partially inspected, based on which an adaption of the MikiWiki environment was conducted. The whole design study included five sessions. For each session the environment had to be prepared.

Semi-structured Interviews

After each design session, the meta-designer conducts follow-up semi structured interviews, for a total of 13 interviews. Open-ended questions are used as we intend to find out what participants think about MikiWiki, their design experiences and the rationale behind their opinions.

The interview questions focus on how MikiWiki supports participants in externalizing and articulating their ideas and requirements on an individual level and on a collaborative level, different design experiences and difficulties of using MikiWiki.

Observation

Each design session lasted approximately 60 minutes. It was divided into three phases: 1) Brainstorming and Collaborative Writing (15 min.). Participants were required to brainstorm RE for Creativity Barometer, to agree on design goals, basic design elements, constrains, and to create a mood-board to illustrate design "look and feel". 2) Sketching Ideas and Collaborative Drawing (15 min). Participants were required to sketch the structure, navigation and components of the application. 3) Designing with the Mockup Environment (30m). Participants could use the mockup environment to finalize the Creativity Barometer interfaces. Although design sessions do not directly relate to each other, certain nuggets were modified in between to support a better RE process.

During the design session, we took observation driven notes with respect to the following aspects: the transition between meta-design, design and use; participants' situated appropriation; how participants with different backgrounds and roles externalize and exchange their ideas, shape their design space to better organize their design flow and design tasks on hand; and how participants brainstorm, articulate and finalize their creative ideas via different nuggets at different design phases.

4.4   Participants

The design sessions involved 11 participants - four female and seven male, aged from 25 to 55 years, and comprising MA, MSc and PhD students as well as associate professors. All the participants are involved in innovation, creativity, CSCW and CSCL related research and are willing to try out new technology. They have some experiences with interdisciplinary creative collaborations, and are used to using different groupware systems. Some participants are directly involved in creativity related research. Every participant has an interdisciplinary focus, ranging from computer science, and usability engineering to sociology, history and political science.

We conducted 5 design sessions, which were organized to involve different types of participants. Group 1 and 2 consisted of two designers; group 3 consisted of two users and two designers from the previous design session; group 4 was made purely of two users; group 5 consisted of one designer and two users. Two participants from group 1 also attended the third design session in order to validate the previous experience and evaluate improvements of the mockup design environment; therefore they were interviewed twice. The second round of interviews focused on whether they noticed any changes to the design environment from their first design session. [In01] to [In13] are used in the text to identify the 13 interviews.

5   Findings: Creativity by Situated Design

In this section, we describe how participants used and appropriated MikiWiki to come up with requirements for *Creativity Barometer* and how meta-designers improved MikiWiki based on situated RE from participants to further support participants' RE and creative in use.

5.1   Support for externalization and communication

A palette of tools: providing simple, small and rich tools is important to support multi-modal creation and different cognitive styles. Small tools allow all the stakeholders to play with, tinker and try use cases and the differentiation of cases in accordance with certain conditions. It is necessary to support participants in exploring solutions and "what-if" scenarios, trying out assumptions to assess requirements continuously. Using MikiWiki with an interactive large screen can be characterized as a 'sandbox for tinkering'. [In02] "*It was quite nice that we didn't jump from tool to tool to do different things. Brainstorming feels more like a different tool, starting from a simple GUI. We just tried what we had there to achieve what we wanted. It really felt like a little playground, when you had quite many possibilities. [...]*"

Fig. 1 Borrowing design elements from the brainstorming stage

For example, fig. 1 illustrates that two participants used various nuggets to externalize and document expectations. Referring to these externalizations on the large screen allowed participants to explain their requirements and design rationale, and to intertwine their perspectives and to foster synergy building. The visualized ideas were a continuous basis for refining and extending them from moment to moment. They also "borrowed" their brainstorming phase robots and statistical image notes directly into the final output phase. Nuggets were therefore used to intertwine their diverse perspectives as well as bridge different design phases.

Notably, two participants had different opinions about the "look and feel" of the barometer interface at the beginning, and they rapidly prototyped a robotic style and a "*Hello Kitty*" pink style (see fig. 1) to express different emotions and feeling with respect to the characteristics of the system to be designed – and consequently to the requirements it will have to meet.

Visualization and externalization: participants used different nuggets to externalize ideas, making tacit knowledge imaginable to others. Fig. 2 demonstrates that one designer used the *sync-imagenote* nugget to search for images from the web to illustrate his flower menu concept and further used the *doodle* nugget to sketch his flower gesture concept. Nuggets provided lightweight means to support each participant to effectively exchange creative ideas and enrich the RE process. "*The good thing with MikiWiki is that it is very wide. It supports different ways of expressing ideas, you have seen that one wants to paint, one wants to use icons, one wants to use photographs, one wants to use text....[In08]."* On the other hand MikiWiki facilitates participants to reach a common understanding by interacting with the concretely available tools and materials. "*It´s fast, you can directly show your ideas, and improve them. If I have an idea and I show it to another person, and then the other person could say, "Yeah this is good or bad, but I think it would be better..."* - the other person can directly show me what he means [In09]."
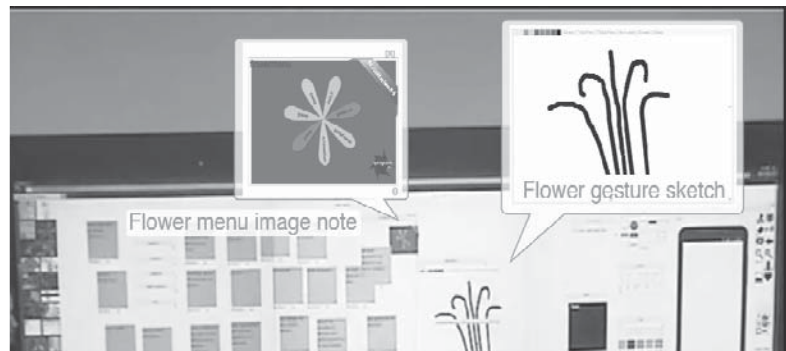
Fig. 2 Visualizing and externalizing concepts

Continuous restructuring: participants were able to act on their design space and redefine it around their specific situated context. As nuggets are independent and loosely coupled, participants could recombine them to better communicate their ideas, to create either a structured design space [In01] or a more chaotic space on the canvas [In03].

One of the designers commented, "*MikiWiki combines everything with everything [In13].*" The interesting point here is that participants were constantly unwittingly creating their design space to better externalize, articulate and share how they envision the requirements for the final system. The flexibility of combining nuggets supported their situated appropriation and adaptation. Nuggets were small and generic enough to be used individually or together to restructure design space [In04] and to achieve new behaviors [In04, In11]. These possibilities of continuous restructuring are a suitable basis for a continuous refinement of visualizations, which can be employed to systematically derive requirements.

Generating stimuli: when participants saw a wide range of icons made available by the meta-designer, they were inspired even if the icons were not directly related to their actual ideas. These items acted as a stimulus for coming up with creative requirements. For instance, in design session 3, designers noticed the audio icon, and subsequently had the idea that audio input should be available. They further reasoned on using voice volume to indicate the rating scale. Introducing unexpected and accidental inputs can foster creativity and simulate unconventional thinking. In particular, the *sync-imagenote* nugget offers easy manipulation with randomness. "*What was quite good was imagenotes [the sync-imagenote nugget]. You could search images from Google. It was mainly for creativity, I think it was cool. […] It's fun to use. It was more to open your mind... [In03]*"

5.2   Moment to Moment

In this meta-design study in the context of co-located meetings, collaborative design becomes an activity within which composition and execution as well as thinking and doing converge in time. It is tightly coupled with certain socio-technical conditions. "*With MikiWiki, you are making ideas, and trying them out at once and in real time…. In one hour, we developed four scenarios, which were quite good ideas [In01].*" Problems are solved without scripted plans or preconceptions of how to do it. Therefore, the decision-making process is situational, i.e. testing and creating on the spot. The temporal dimension is compressed from several connected time spans to

moment-to-moment simultaneous decisions. With respect to RE, the setting neither focuses on the systematical scripts of creativity techniques as for example described by Briggs de Vreede (2006) nor does it follow certain phase models as proposed by Osborn and Parnes (Kaufman and Sternberg 2006) or Shneiderman (2002). By contrast, we employed electronic media to support a much more spontaneous approach since the continuous recording of the results of the session allows the participants a succeeding analysis of their contribution on which further elaboration can be based.

5.3 Meta-design

The distinctive aspect of MikiWiki is that different design activities occur within the same system. After each design cycle, in accordance with the participants' feedback and meta-designers' observation, nuggets were modified or created for the next design session to better support the design process. Consequently, the nuggets were constantly a subject of design and reframed by designers' and users' creative contributions. In contrast to the traditional software development approach, designing everything in advance, "*Design Now*" for the meta-designers is rather designing in the moment.

For the meta-designer, MikiWiki strongly supported a situated design-in-use option making it both possible and easy to adapt the design space from session to session. It is through this cyclical process that meta-designers, designers and users enhanced their mutual understanding by interacting with the concretely available tools and materials. As an example, meta-designers were able to improve the *doodle* nugget step by step e.g. modifying menu, adding auto-saving function, providing animation function etc. and through the whole design sessions. This also implies the progression of design sessions and the co-evolution that took place between users, designers and meta-designers.

The co-located approach is particularly valuable in investigating meta-design support, since emergent social-technical issues, user behavior patterns and dynamic interactions between various roles can be directly observed, influenced and recorded. Thus, meta-designers are able to get instant feedback and improve MikiWiki at the meta-design level in an agile manner.

The transition between meta-design, design and use in MikiWiki supports iterative design processes, thus converging towards requirements in contrast to the traditional systematic description of functional specifications.

5.4 Different levels of creativity

During the session we observed different levels of creativity due to different activities, namely opportunistic programming as bricolage at the meta-design level, and creativity-in-use at the design and the use level, in that designers and users invent their own ways to use MikiWiki which are not envisioned by meta-designers. Nevertheless different levels of creativity strongly influence one another:

- Meta-designer level: constructing design environments is an activity occurring at the meta-design level, in that the meta-designer sets up the initial design environment for the design session and constantly evolves it opportunistically to cope with emergent socio-technical issues without needing to change server-side code.

- Designer level: design environments support creativity at the design and the use level, in that participants continuously adapt nuggets to form a design space in order to perform their design tasks at that moment.
- User level: participants use the tailored design space at different phases to externalize their thoughts on the fly.

A meta-design approach is essential in supporting different levels of creativity: creative design from meta-designers and creative appropriation from designers and users triggering further creative meta-design.

The validity of the empirical findings has certain limitations as meta-design normally covers a much longer period than was observable within the case study. Ongoing empirical investigation of a meta-design approach enabled RE should be conducted. In addition, how these requirements of *Creativity Barometer* derived from design sessions are implemented and adapted in practice needs further study.

6   Conclusions

We did not follow a scripted approach of applying creativity techniques for requirements development in systematically facilitated group meetings as it is pursued – for instance - by (Jones and Maiden 2005). Instead "*Design Now*" summarizes our approach from the following aspects:

1) The RE process is situational, testing and creating on the spot. Decisions are made collectively, contingent and from moment to moment.

2) With respect to the meta-design system, new requirements emerge in time and are tightly coupled with current conditions, which have little to do with scripted plans and can be collected and implemented at use time.

3) Situatedness is a decisive characteristic of situations where people "dive" into interplay between drafting of a software solution on the one hand and understanding their needs and expectations on the other hand. Creativity can benefit from situatedness, as underpinned by Csikszentmihalyi's concept of flow (Csikszentmihalyi 1996). Within a flow, people's attention is completely attached to their goals e.g. by intensively interacting with artifacts.

Additionally, this study suggests that a more visual-based approach can be further explored in supporting creative and collaborative drafting of the features of a software system. As demonstrated in MikiWiki, stakeholders with a kind of artifacts which do not focus on textual descriptions of requirements but can immediately 'talk back' (Fischer et al. 2004) from the very beginning. Creating visual descriptions of what users do expect or need, can be considered as a starting point for deriving specifications for functional requirements for instance by identifying:

- The use cases and the differentiation of cases in accordance with certain conditions, e.g. whether user wants to be pushed by the barometer or not;
- The larger context of the application as addressed in contextual design (Beyer and Holtzblatt 1998), e. g. to avoid interruptive conversations;
- The experiences which might be supported by using the tool to be developed, e.g. by the style of the interface;
- The metaphors on which the user interface should be based;
- The dialogue steps which are presented by different states (i.e. mock ups) of the interface;

- The data which has to be included in a data model;
- The control flow by which the dialogue with the system should be guided.

It turns out that the closely intertwined cycles of meta-design and design with MikiWiki do not aim at completed products or well-defined requirements. In contrast, the design outcome always makes sense with respect to a concrete situation and mainly helps to increase the stakeholders' understanding of what they expect according to their different perspectives. This is driven by the participants following their inclinations and design instincts in the pursuit of their evolving goals as it is typical for wicked problems or for the relevance of mess finding (Osborn and Parnes; cf. Kaufman and Sternberg 2006).

References

Avison DE, Lau F, Myers MD, Nielsen PA (1999) Action research. Commun ACM 42 (1):94-97. doi:10.1145/291469.291479

Beyer H, Holtzblatt K (1998) Contextual Design: Defining Customer-Centered Systems. Morgan Kaufmann Publishers, San Francisco

Blum F (1955) Action research—a scientific approach? Philosophy of Science 22 (1):1-7

Bourguin G, Derycke A, Tarby J-C (2001) Beyond the interfaces, Co-evolution inside Interactive Systems: A proposal founded on the Activity Theory. Paper presented at the IHM-HCI 2001, Toulouse.

Csikszentmihalyi M (1996) Creativity — Flow and the Psychology of Discovery and Invention. HarperCollins Publishers, New York, NY

De Vreede G-J, Kolfschoten GL, Briggs RO (2006) ThinkLets: a collaboration engineering pattern language. Int J Comput Appl Technol 25 (2/3):140-154.

Doerry E (1995) Evaluating distributed environments based on communicative efficacy. Paper presented at the Conference companion on Human factors in computing systems, Denver, Colorado, United States

Fischer G, Giaccardi E, Ye Y, Sutcliffe AG, Mehandjiev N (2004) Meta-Design: A Manifesto for End-User Development. Communications of the ACM 47 (9):33-37

Fischer G, Herrmann T (2011) Sociotechnical Systems: A Meta-Design Perspective. International Journal for Sociotechnology and Knowledge Development 3 (1):1-33

Herrmann, T. (2009), Systems Design with the Socio-Technical Walkthrough, in Brian Whitworth & Aldo de Moor, eds ., 'Handbook of Research on Socio-Technical Design and Social Networking Systems', IGI GLobal, Hershey, PA, USA, pp. 336-351.

Herrmann T (2010) Support of Collaborative Creativity for co-located Meetings. In: Randall DS, Pascal (ed) From CSCW to Web 2.0: European Developments in Collaborative Design. Computer Supported Cooperative Work. Springer London, London, pp 65-95.

Herrmann T, Carell A, Nierhoff J (2011) Creativity barometer: an approach for continuing micro surveys to explore the dynamics of organization's creativity climates. Paper presented at the Proceedings of the 8th ACM conference on Creativity and cognition, Atlanta, Georgia, USA

Jones S, Maiden NAM (2005) RESCUE: An Integrated Method for Specifying Requirements for Complex Sociotechnical Systems. In: Maté J SA (ed) Requirements Engineering for Sociotechnical Systems. Idea Group Publishing, London, pp 245–265

Kaufman J, Sternberg R (2006) The International Handbook of Creativity. Cambridge University Press, New York

Lévi-Strauss C (1968) The Savage Mind. University Of Chicago Press

Müller M, Pfeifer R (1997) Developing effective computer systems supporting knowledge-intensive work: situated design in a large paper mill. In: Cases on information technology management in modern organizations. IGI Publishing, pp 225-249

Peffers K, Tuunanen T, Rothenberger M, Chatterjee S (2007) A Design Science Research Methodology for Information Systems Research. J Manage Inf Syst 24 (3):45-77.

Pfeifer R, Rademakers P (1991) Situated Adaptive Design: Toward a New Methodology for Knowledge Systems Development. Paper presented at the Verteilte Künstliche Intelligenz und kooperatives Arbeiten, 4. Internationaler GI-Kongress Wissensbasierte Systeme

Resnick M, Myers B, Nakakoji K, Shneiderman B, Randy Pausch, Selker T, Eisenberg M (2005) Design Principles for Tools to Support Creative Thinking. IJHCI, 36 edn.,

Shneiderman B (2002) Creativity support tools. Commun ACM 45 (10):116-120.

Suchman LA (1985) Plans and Situated Actions: The Problem of Human-Machine Communication. Xerox Palo Alto Research Center, Palo Alto, CA

Zhu L (2011) Cultivating collaborative design: design for evolution. In Procedings of the Second Conference on Creativity and Innovation in Design (DESIRE '11). ACM, New York, NY, USA, 255-266.

'Pictionades': Enhancing Stakeholders' Awareness about Issues in Requirements Communication

Deepti Savio and Anitha P.C.

Corporate Research & Technologies, Siemens Information Systems Ltd., Bangalore, India
`{deepti.savio,pc.anitha}@siemens.com`

Abstract. The various issues involved in communicating requirements across multiple stakeholders and stakeholder groups have been well documented in literature and in experience reports. Despite this, however, most stakeholders involved in a project seem largely unaware of what the potential consequences of these issues can be. The manner in which stakeholders communicate requirements to each other affects the subsequent requirements management activities, and has a direct impact on the final form and scope of the stated requirement. Here, we discuss the approach of using a combination of two popular group games to convey to stakeholders without a requirements engineering background the realities that underlie the communication of requirements across multiple points. We then discuss the results of applying an adaptation of this technique in a real world project.

Keywords: requirements communication, stakeholders, stakeholder groups

1 Introduction

Communicating requirements is one of the most crucial aspects of managing requirements throughout a project. The manner in which requirements are captured plays a key role in determining if they can be read, analyzed, re-written if necessary, and validated [1]. Stakeholders without a requirements engineering (RE) background often do not realize the impact of the consequences that requirements communication issues can give rise to, while dealing with requirements at various phases of the project life cycle. Communicating requirements without sufficient domain knowledge or understanding of the context of the requirement, dilution of information, floating stakeholders who work through multiple projects, partial and conflicting stakeholder views [2] and so on are some of the causes of misinterpreting requirements while conveying them. Although some stakeholders may be aware of these problems, they may not always grasp the full extent of the potential implications of these issues.

Convincing stakeholders about the ground realities of these concerns and their possible ramifications, as well as helping them understand and appreciate that other stakeholders have differing points of view is the first step in reducing the chances of ambiguity and uncertainty that are often reflected in requirements.

Therefore, in order to:

- help raise awareness of the issues described above among stakeholders who do not have an RE background,
- convey the significance of the impact of these issues in a creative and enjoyable manner, thus eliminating the need for stakeholders to have to go through extensive documentation, and
- ensure better retention of these messages through appropriate analogies,

we carried out an exercise involving a combination of two popular group games, Pictionary and Charades – *Pictionades* – to demonstrate that requirements communication has several inherent problems, especially in distributed project settings [3], and that these concerns must be considered while making requirements–related decisions throughout the project life cycle.

We decided to capitalize on the efficiency and ability of simple group games which can be played at the workplace to drive messages whose importance is otherwise often underestimated. Several games for eliciting requirements have been used in workshops and in industry, such as [4], [5], [6] and [7]. The use of Pictionary for working with students to teach requirements analysis is discussed in [8]. However, there are fewer experiences of using of games for subsequent requirement management activities after elicitation. The main objective of this exercise is to help the participants appreciate, from the perspectives of other stakeholders, the difficulties that invariably creep in during the communication of requirements. The game is structured in such a way that the participants are able to easily relate the outcomes with their own experiences in communicating requirements [9]. We made use of role plays [10] in our technique in order to stress further on the lessons that we wished to relay. We report the outcomes of this experiment and discuss an application of a variant of this game in a real project.

2   Pictionades: Game Setting and Play

Project managers in any project often play a pivotal role in decision making and conflict resolution among stakeholders. If they are equipped with:

- the realization that requirements communication issues are many, and that their possible consequences could seriously affect the overall objective of the project
- the ability to understand matters from the perspectives of the other stakeholders involved, and thus endeavor to efficiently reduce conflicts and create a good rapport among stakeholders
- the means to ensure, as far as possible, that all teams work with a clear, unambiguous set of requirements at all times,

then, this awareness would enable them to make and take better substantiated and well informed decisions during the course of the project. We decided to try out a trial version of Pictionades during a training program for project managers (hereafter referred to as 'PMs') in our organization, to see if the approach would work.

2.1 Pictionades set-up

The participants of the game were 9 PMs, having several years of experience on a wide range of industry projects. We divided them into two teams - team 'A', comprising three PMs, and team 'B' with six PMs. Each person in both teams assumed one of three roles – the artist, the actor or the interpreter. A few high level requirements for two products were given only to the artists from each team, who read the information written on a slip of paper. The artist stood at the board, the actor at the opposite end of the room, and the interpreter was seated in the middle, as shown in Figures 1a and 1b.
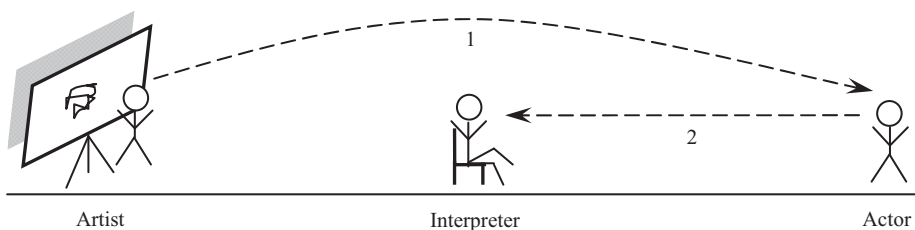
Fig. 1a. Team A: Single Stakeholder per group
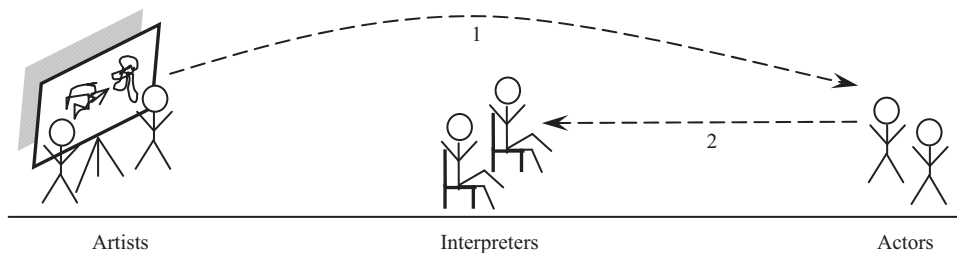
Fig. 1b. Team B: Two Stakeholders per group

The artist faced the actor and tried to draw out the given information on the board, without talking, and without the use of any written language. The actor, at the other end of the room, observed the drawings on the board, and, facing the interpreter (who was seated between him and the artist), acted out what he inferred from the drawings on the board to the interpreter, again, without talking. The interpreter, who had his back to the artist and could therefore see only the actor's actions (please see Figures 1a and 1b), wrote down his interpretations of the actions. He was allowed to ask questions to the actor, to which the actor could reply only with a nod or shake of his head. Note that there was no direct communication in any form between the artist(s) and the interpreter(s) of either team.

We took a few high level requirements from the end user's perspective, for two example products - an online book shopping portal and a smart phone. Since we felt that the example of the smart phone was a bit more complex than that of the online

book portal, Team A, who were given the online bookshop product, had one member in each role, and Team B, who were given the smart phone product had two members in each role.

The high level requirements for both examples given to the artists of both teams are listed below in Table 1.1:

	High Level User Requirement 1	High Level User Requirement 2	High Level User Requirement 3	High Level User Requirement 4
Team A: Online book shopping portal	"I want to shop online for all kinds of books"	"I want two payment options – online, using my card, and/or cash payment when the books are delivered to my place"	"I want the site to remember my preferences when I use it next, and show me books that I'd be interested in"	"I want to search for the books I'm looking for in all retail outlets in my area via the site"
Team B: Smart phone	"I want to sync all my facebook, outlook and google contacts"	"I want text to speech conversion so that I can listen to articles or read the news paper when I'm driving"	"I want to be able to record a video and simultaneously stream it online using my phone"	"I want a mobile TV feature in my phone so that I can watch my favourite shows while on the move"

Table 1.1. Initial user requirements list for an online book shopping portal and a smart phone - given to only the artists of each team

We monitored both groups as they worked in parallel in the two halves of the room. The participants did the best they could, and, at the end of their allotted twenty minutes, the interpreters of both teams managed to come up with the following snippets of information:

	Point 1	Point 2	Point 3	Point 4	Point 5
Team A's Interpreter	"Screen, and general UI on screen"	"Buttons like ok, cancel, etc are available"	"Text box to search for something"	"Search results"	"Google"
Team B's Interpreters	"Connector with 3 interfaces"	"For email, facebook and outlook"	"Contacts list"	"While driving it is possible"	"Look at contacts and make calls"

Table 1.2. Results at the end of play

2.2   Rationale for Game Setup

When the results at the end were collected and divulged to all members of both teams, it was discovered that the actors and interpreters had roughly gauged only a

few parts of some of the requirements, without having an overall idea of what the product was. The objectives of structuring the game in this fashion are listed below:

No/limited verbal communication:

Language is often a problem between stakeholders who may be not only from different countries, but also from different regions within a country. By eliminating verbal communication between the artist and the actor, we wished to highlight the issue of how two stakeholders would communicate if they had limited knowledge of each other's language(s), or no language in common through which they could communicate. This situation compelled the participants to consider a combination of alternate approaches of communication, such as the emphasized use of body language, gestures and expressions, along with pictorially representing the given information.

Disconnected communication link between the artist and the interpreter:

In real world scenarios, there is often no direct communication between stakeholder groups. For example, the end-user of a system and the system architect generally do not communicate directly with each other - the market research team (or the RE team) serves as the interfacing link between these two stakeholder groups. Furthermore, there are instances of numerous groups of co-located stakeholders who may or may not be in touch with each other. Hence, utmost care must be taken to ensure that stakeholders' needs are identified and captured as concisely as possible, so that the intended information is transmitted, received, and subsequently relayed from stakeholder group to stakeholder group.

Distance between stakeholder groups and time constraints:

We ensured that the artist, actor and interpreter were positioned far away from each other in the room to signify that face to face discussions with stakeholder groups may not always be possible and to underscore the fact that distance may impede communication. Despite the difficulties encountered, the participants were obliged to try and complete their mission while working within the constraints in the allotted interval of time, as is the case in reality.

2.3   Highlighted Communication Issues:

The lessons which we wished to convey to the PMs through Pictionades are listed below:

From the stakeholder perspective:

- It is critical that stakeholders understand and appreciate differences in other stakeholders' 1) assimilation of context-based understanding, 2) domain knowledge, 3) language, and 4) communication skills.
- Each stakeholder in the communication chain will interpret an incoming requirement in a particular form, in his or her own way, depending on his or her

background knowledge and level of domain understanding, and consequently relay it another form, in his or her own way to the next stakeholder, as shown in Figure 2. (This is analogous to the game 'Chinese Whispers', which is often what happens in real world instances of communication!)
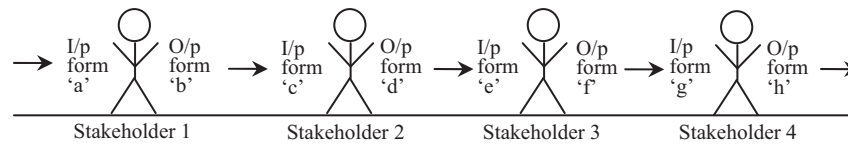
Fig. 2. Input and output form of a single requirement to and from each stakeholder in a simple, linear communication chain (In reality, there is a complex network of stakeholder groups who may or may not be in touch with other stakeholder groups).

- Multiple stakeholders within a group (for example, several system developers and testers in the development and test teams respectively) will result in a greater exchange of ideas, but stakeholders within a group may have differing opinions.
- The presence of many such stakeholder groups along the communication network may often result in further misinterpretation of requirements – hence leading to the observation that requirement decisions must be taken as a group and then communicated. Here, the importance of setting up single points of contacts for each stakeholder group was identified.

From the communication techniques perspective:

- The overall system or product and its purpose must be described as clearly as possible before communicating its features.
- Each stakeholder has his own limitations in communication techniques, and these limitations may cause constraints in communication. Furthermore, additional such constraints may be encountered at any point – stakeholders must gradually learn to work within and around these constraints.
- Several rounds of communication may be required to obtain clarity on a single point. This point was demonstrated when we observed the participants trying to refine their articulation of the information that they wanted to communicate, and provide more intelligibility on expression techniques, when it was observed that the recipient of the information hadn't fully grasped the information.
- Establishing well understood terminology that can be understood irrespective of stakeholders' backgrounds is an effective means in which communication can be made more effective. This point was observed when we saw the participants instinctively making use of hand signals such as a raised palm for 'stop and start over', a quick shake of the hand/head for 'no, incorrect', a thumbs up sign for 'right' and so on, when they realized that the other participants easily understood these gestures.
- It is important to communicate what the system is *not* supposed to do, in addition to what features and functions it must exhibit. This would help in the formulation of both positive and negative use-case scenarios of the system. This learning is

indirectly helpful for stakeholders in determining what components should go into the system, interface, environment and domain [11]. We observed our participants sub-consciously trying to delineate which entities fit into what category, while thinking of haves and not-haves for the product, thereby carrying out an intuitive scope analysis without actually being aware of what they were doing.

- The importance of continuous validation and feedback on communicated items as and when possible cannot be underestimated. This stresses on the fact that the only way of obtaining good quality requirements is by reviewing, reviewing, and reviewing.
- The articulation and communication of non-functional requirements of the product across distributed stakeholders would be even more difficult, given constraints in communication techniques.

3   Applying a modification of Pictionades to a real world project

We now discuss the application of an adaptation of this technique to a real world project. Project X is currently underway for the development of a software system for the management of an industrial automation plant, and is still at an initial stage. Before we set about discussing the requirements for the system, we asked the end user representatives to communicate to us their understanding of the overall context of the system, as well as the layout of the plant through pictures. The plant turned out to be a complex combination of several large scale systems, operations and processes working together. Physical components such as 'stackers', which deposit raw materials into organized, measured stockpiles, and 'reclaimers', which collect pre-determined amounts of the material from specified stockpiles, as well as logical entities such as the processes and workflows followed to transport these materials from one place to another were identified.

Nguyen and Shanks suggest that creativity techniques be enmeshed during RE activities from the product, process, domain, people and context perspectives [12]. Following this thread, we made use of personas (finding out, describing and assigning realistic personalities to each of the stakeholder groups for the software system) during requirements discussions with the various end user representatives for Project X. Each member of the end-user representative group was assigned a particular persona and then asked to draw out his mental picture of what the system would look like, from the perspective of his persona. We kept the personas as distinct and diverse as possible, in order to capture the gamut of the possible types of people who would operate the system, and the various conditions in which the system would be operated.

We collected the pictures together and mapped out their common features. This helped us in further understanding the context, as well as the surrounding environment of the system. We were able to determine that there were several categories of end users for the system, and were encouraged to think from each of the groups' perspectives, and visualize their notion of the look and feel of the system. Since text based representations were kept to a minimum, discussion times were cut down considerably – we were able to come to a quick consensus – and visual clarity

was provided on most of the interfaces. Due to this, we were able to elicit and communicate several requirements for the user interface of the software system which we may have otherwise overlooked. We were also able to determine use case scenarios, and develop a context diagram with actors and end users from the pictures.

4   Conclusion

Here, we describe the experiences of conducting a combination of the games Pictionary and Charades among project managers in the hope that it would aid them in making better and more well–informed requirements related decisions during the course of project execution. The usage of drawing out information from the perspectives of other stakeholders was then tried out for a real world project, with encouraging results. The game was played by project managers, with the intention of helping them understand and appreciate the inherent difficulties involved in communicating requirements. Pictionades is easy to play, and successfully serves as an ice-breaker among stakeholders who are unfamiliar with each other, while additionally creating a light and lively atmosphere to work in. The game can be played by all stakeholder groups in a project, and would encourage each role to think from the perspective of the other participating roles. Since the game doesn't take much time, and can be carried out using readily available office stationery and material, it can be conducted during group discussions, training sessions, and so on. In the future, we would like to see how Pictionades may be extended to:

- other types of requirements, apart from functional requirements – such as non functional, environment, quality, performance related requirements, and so on.
- the other requirements management activities in the life cycle of a requirement – such as resolving and closing issues quickly in change control board meetings
- various development methodologies, such as the agile family of methodologies

The scope of the game could also be increased in the literal sense by having several groups of artists, actors, and interpreters, or even creating more roles. Several such groups could form a longer communication chain. Additionally, more complexity (and a sense of reality) could be added by breaking the linear arrangement of stakeholder chains and creating a more complex network that is spread out across the room.

An advantage of using the Pictionades approach for the purpose of helping participants appreciate and assimilate the impact of issues encountered in requirements communication is that the technique stresses on using multiple forms of communication, apart from just verbal exchange of information, thereby pushing players to think outside the box as the game progresses, and come up with unconventional, but effective solution possibilities. Furthermore, Pictionades is designed and set up in such a manner that the players are easily able to apply the experiences and outcomes of the game in the context of requirements engineering [9].

References

1. Nusibeh, B., Easterbrook, E.: Requirements Engineering: A Roadmap. In: Future of Software Engineering (FOSE '00), pp. 35–46, ACM, New York (2000)
2. van Lamsweerde, A.: Requirements Engineering: from Craft to Discipline. In: 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering (SIGSOFT '08/FSE-16), pp. 238–249, ACM, New York (2008)
3. Herbsleb, J. D.: Global Software Engineering: The Future of Socio-technical Coordination. In: Future of Software Engineering (FOSE '07) pp.188–198, IEEE Computer Society, Washington (2007)
4. Millard, N., Lynch, P., Tracey, K.: Child's play: using techniques developed to elicit requirements from children with adults. In: Third IEEE International Conference on Requirements Engineering, pp. 66–73, IEEE Computer Society, Washington (1998)
5. Maiden, N., Gizikis, A., Robertson, S.: Provoking Creativity: Imagine What Your Requirements Could Be Like. IEEE Software, vol. 21, pp. 68–75, IEEE Computer Society Press, Los Alamitos (2004)
6. Hainey, T., Connolly, T.: Evaluating Games-Based Learning. International Journal of Virtual and Personal Learning Environments (IJVPLE) 1, pp. 57–71, IRMA, Hershey (2010)
7. Boulila, N., Hoffmann, A., Herrmann, A.: Using Storytelling to Record Requirements: Elements for an Effective Requirements Elicitation Approach. In: Fourth International Workshop on Multimedia and Enjoyable Requirements Engineering (MERE '11), pp. 9–16, IEEE Computer Society Press, Los Alamitos (2011)
8. Beatty, J., Alexander, M.: Games-Based Requirements Engineering Training: An Initial Experience Report. In: 16th IEEE International Requirements Engineering Conference, pp. 211–216, IEEE Computer Society, Washington (2008)
9. Alexander, M., Beatty, J.: Effective Design and Use of Requirements Engineering Training Games. In: Third International Workshop on Requirements Engineering Education and Training (REET '08), pp. 18–21, IEEE Computer Society Press, Los Alamitos (2008)
10. Zowghi, D., Paryani, S.: Teaching Requirements Engineering through Role Playing: Lessons Learnt. In: Eleventh IEEE International Conference on Requirements Engineering, pp. 233–241, IEEE Computer Society, Washington (2003)
11. Gunter, C.A., Gunter, E.L., Jackson, M., Zave, P.: A Reference Model for Requirements and Specifications. IEEE Software 17, pp. 37–43 (2000)
12. Nguyen, L., Shanks, G.: A framework for understanding Creativity in Requirements Engineering. J. Inf. Softw. Technol. 51, 655–662 (2009)

Requirements Analysis for Multimedia Interactive Informative Systems: A Metamodelling Approach

Sylviane Levy and Fernando Gamboa

Universidad Nacional Autónoma de México

sylviane@unam.mx, fernando_gamboa@cuaed.unam.mx

Abstract. Multimedia Interactive Informative Systems (MIIS) are software applications resulting from the convergence of multiples technologies such as audiovisual, computing and communication. As other mass media, they aim to transmit information to a large, diverse and dispersed public. Capturing requirements for those applications is one of the most difficult problem since final users are non-captive. An approach for designing a MIIS should follow the same methodology used for scriptwriting. As of now there are no techniques for transforming a script into requirements that can be used by a software development team. In this paper, we propose a metamodel to be used as a domain specific language when designing a MIIS and define new communicative quality attributes to complete ISO/IEC 25010 quality in use model. A new approach is proposed, based on analyzing class diagram through quality attributes to establish MIIS requirements and applying it through a case study.

Keywords: Interactive systems, Multimedia, Requirements analysis, Metamodel, quality-oriented design

1   Introduction

In this study, we define Multimedia Interactive Informative Systems (MIIS) as software systems that result from the convergence of multiple technologies, for example the combination of computational, audiovisual and communication technologies. The purpose of MIIS is to diffuse information to a large, dispersed, diverse, and non-captive public. An example of a MIIS would be a website whose goal is to transmit information, such as an online newspaper or a cultural website. A kiosk in a museum, cultural CD-ROMS, and many applications for mobile phones and tablets can also be considered as MIIS.

As with other communication media, such as cinema and television, whether or not the MIIS fulfils its goals depends largely on quality characteristics related to the communication between the transmitter and the receiver. In this case, the qualities of communication between the MIIS and its final users are essential, and the success of the MIIS depends on the user finding these systems *interesting, informative, and credible*. However, these criteria are not included among the characteristics of quality

models used for software such as the ISO/IEC norm 25010 [1] and its predecessor ISO/IEC 9126.

These types of needs are difficult to express, and even more difficult to translate into system requirements. But if they are not satisfied, the user can discard the application with the same freedom that she or he can change the channel of the television or leave a cinema when the film is disappointing, rendering the system essentially useless.

It is not accidental that the existing models of quality do not consider these types of characteristics. Software applications traditionally help captive users to solve concrete problems, and therefore the problem of retaining interest does not exist. The MIIS's user, on the other hand, is generally non-captive: he or she has taken a voluntary personal decision to acquire the information, and is not bound to use the system by any degree of necessity.

Traditionally, the requirements of a computer system are established based on an analysis of the needs expressed by the final users. In the case of MIIS, where the objective is to transmit information, the end users cannot easily decide on the content of the application. This therefore generally requires the participation of content experts who then decide what information should be contained in the system, not the users. [2]

The above observations explain why it is necessary to explore new methodologies to establish the requirements of an MIIS, as well as to find new attributes of quality that are more appropriate to the transmission of knowledge to non-captive users.

This study proposes a methodology that allows the establishment of the requirements of an MIIS. We propose a metamodel of MIIS to be used as a domain specific language used to design the system. In this way, the model of quality in use based on ISO/IEC 25010 will be enriched with new characteristics of quality that will allow the completion of an analysis of requirements for the system.

This paper is organized as follows: section 2 is devoted to the background of our work through the study of existing methodologies; section 3 presents Multimedia Interactive Informative Systems (MIIS) as a metamodel; section 4 defines the expected qualities for MIIS; a case study is detailed in section 5 and section 6 is the conclusion.

2   Background

Studies covering the development of multimedia have split along two general lines:

- The first line derives from the field of communication, where the authors concentrate on the creation of a new communication medium. These works are based on a development process that is strongly influenced by that used for making films or videos.
- The second line derives from the design and development of conventional software, and is focused on Requirement Engineering and on the resolution of problems linked to the handling of multiple media, but leaves aside the communicational problem.

2.1 Communication perspective

In general, the works that have come from the field of communication studies are written as scholarly manuals and use as a conceptual and development methodology from other media, such as cinema or television.

One of the most mature of these works is by Friedmann, who proposes a process of analysis and scriptwriting for different means of communication, which he calls *visual media*, which include traditional media but also websites, educational and training software applications, information kiosks, and videogames [3]. Friedmann establishes an analytical process whose goal is to yield a *creative concept* of visual media on the basis of which the script is written.

Nonetheless, even in the case of interactive systems, the screenplay is written in a style strongly influenced by the cinematographic industry, given that: "Books, movies, television, theatre—all imply the creation of specific documents that establish formats that the interactive industry does not have." [3] In particular, the screenplay does not specify how it can be converted to the requirements of a system that must be understood, interpreted, and developed by a multidisciplinary team of designers, communications specialists, and programmers, among others.

This study will use Friedmann's framework to analyze the field of MIIS and to establish a concept for application. .

2.2 Software perspective

In the field of computing, there are few studies which consider the transmission of information as the main purpose of the system. Among the most original studies are those from Bolchini, who invented the term "infosuasive" to define Web applications that are both informative and persuasive, such as those applications that "intend to transmit information and have the objective (declared or not) of influencing the opinion of users, their attitudes and conduct." [4]. Bolchini bases his work on the Objective-Centered Design Cooper's methodology [5], centering his analysis on the need to meet the objectives of different users and introducing *communication objectives* as elements of analysis for the design of Web applications. One of his contributions was to identify the need to carry out and to document a communication analysis to build website requirements, such as *content, layout, architecture, and navigation.*

Another work which considers new types of requirements is that of Davide Callele [6], who studies video games in particular and has introduced the concept of *emotional requirements*. These analyze and document the emotions that are desired to be transmitted to the users/players. On the basis of these emotional requirements, he establishes functional and non-functional requirements. The concepts presented in these works will allows us to analyze the requirements of MIIS through qualitative criteria such as interestingness, informativeness and credibility.

There are a variety of techniques to elicit requirements [7], and these generally involve the final users of the system and stakeholders. But they are very few studies that consider requirements that are not elicited by users or stakeholders, which, as noted above, is the case of MIIS. Among these few studies, in [8] it is suggested a methodology based on the analysis of a metamodel using workshops in which different members of the team participate. In [9] metamodels are used as a "medium

for integrating and customizing Requirements Engineering techniques" and in [10], the role of a quality metamodel is discussed.

Based on these studies, we propose the use of a metamodel to represent the system and derive its requirements through the analysis of attributes of quality.

3 Multimedia Interactive Informative Systems (MIIS)

3.1 Definition

As mentioned, MIISs result from the convergence of multiple technologies, with the objective of transmitting information to a wide, diverse, and dispersed audience, and are composed of different multimedia components (text, images, video, animation, and audio). The interactivity means that the system allows, and in fact requires, the active participation of the user in order to fulfill the goal of transmitting information.

The development of MIISs thus varies from that of conventional software in the following ways: communicational problems have to be resolved, different users use the systems, different data enters the system, and the contexts in which the information is used and the technologies developed differ.

3.2 MIIS Components

The existing literature contains a number of proposals to describe multimedia applications created to transmit information. One of the classifications that describes best the composition of a multimedia system is found in [11] where the essential functions described above are illustrated in the "form of circles representing the three components that define a balanced digital project: visualization, interactivity, and content". Based on this description, we will propose a metamodel to describe a MIIS.

3.3 MIIS Metamodel

We propose a MIIS Metamodel composed of three main meta classes, Window, Content, and Scene. (Fig. 1)

- **Window** represents the architecture of the system, which allows navigation between the interfaces.
- **Content** represents the information to be transmitted. It is displayed by different media, which could be simply audiovisual or also interactive.
- **Scene** represents all the audiovisual elements of the interfaces that have the purpose of drawing the user into the application.

From the above, it is clear that what the user can see, hear and interact with, are the media and interactive media:

- **Media** can represent information, contextualize content, have an esthetic, emotional, or functional purpose, or can be interactive visual elements. Media are

made by classes of text, images, sounds, videos, 2D and 3D animations and interactive media.

- **Interactive media** represent the different functions which allow the user to interact with the system. They are formed by any type of interactive component such as the display of media, 3D immersion, simulations, hypertext, tests, queries, etc.
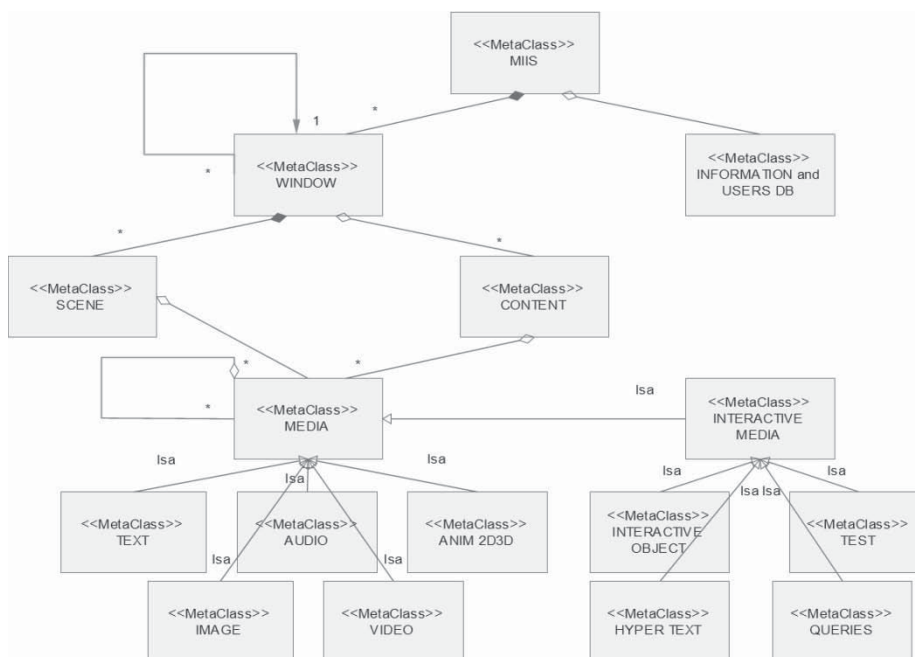
Fig. 1. MIIS Metamodel

4   MIIS Quality

4.1   Quality of a software

Among the models of quality that are most often used to evaluate software is the norm ISO 25010 which substitutes the well known ISO 9126 for the models of quality of software [1] and ISO 25012 [14] for the quality of data.

Among the characteristics of the three quality models, none of them address the communicational aspects of the software, which is to say the quality of the communication that is established with the user. In particular, aspects such as *interestingness, informativeness, and credibility* are not considered in these models.

4.2 Quality of a MIIS

Definition of Communicational quality characteristics.
If a system of quality is to be characterized by its level of communication, the model of quality in use described in [1], should be complemented by new characteristics, whose goal is to evaluate the quality of communication between the system and its users. Three characteristics are introduced as follow:

- *Interestingness* is the capacity of the system or its components to attract the user and maintain his or her interest and attention for a minimum time in a specific context.
- *Informativeness* is the capacity of the system or its components to inform the user with relevance, in a structured way and in a specific context.
- *Credibility* is the capacity of the system or its components to allow the user to feel involved in a credible and authentic environment in a specific context. It allows the user to believe in the veracity of the information that is being transmitted.

Evaluation of Communicational Quality Characteristics.
The evaluation of the characteristics of communication of an MIIS is used generally to evaluate an application under development, typically a prototype. The goal is to assess the impact of its characteristics on the user before the application is completely developed.

The evaluation is carried out across a number of different attributes, applied to a series of components of the application, according to the selected strategy. If one wishes to isolate those components that are not fulfilling their objectives, it is necessary to evaluate the satisfaction of a quality attribute by components. As seen in table 1, it is possible to establish metrics that allow the evaluation of each of the main meta classes. In cases where a questionnaire is used, it is necessary to associate a quality scale to each.

Table 1. MIIS metrics for quality attributes

Quality characteristics	Content quality attribute	Scene quality attribute	Window quality attribute
Interestingness	Time spent by users	Questionnaire about interest of the scenes	• Number of windows visited • Time spent in each Window
Informativeness	• Number of block content • Structure of information • Relevant • Actual • Questionnaire about information retained by users	Questionnaire about Information contained in scenes	
Credibility	• Number of references	Questionnaire about credibility of scenes	

To achieve the quality of communication required, the components of the system must be designed with that intention, which is to say, they must be the product of an analysis of requirements that considers the communication purpose of the application.

5   MIIS Requirements

According to [13], "Quality is a feature that is 'built' in an information system and not added afterwards." In this sense, if we accept that a MIIS should be interesting, informative and credible, those quality attributes have to be taken in consideration as part of the requirements analysis.

Unlike traditional software, which captures the requirements of the final users, in the case of MIIS, the requirements are established based on a creative concept, which is the product of a communicational analysis.

The purpose of the creative concept is to "solve the communication problem, reach the target audience, achieve the objective, embody the strategy, provide the content of the application, and show how it will work". [3] But, at the initial stage it is just an idea that needs to be translated into requirements that a development team is able to understand and execute in a creative way.

In order to establish the requirements of the MIIS, the first step consists of creating a model of the application based upon the concept. As each class inherits some of the MIIS quality attributes, it plays a role in responding to the needs of informing, interesting, and convincing the user. It is then possible to elicit the requirements of each component of the application through the analysis of the attributes of quality.

5.1   Case Study modeling

In order to illustrate the above, we will apply the methodology to a study case: "Pierre y la Coatlicue", a system developed to motivate students of Spanish as a foreign language through, at the same time, learning and understanding Mexican culture.

The system is aimed at a large public around the world. Even if the students can be considered captive in the study of the language, they are not captive in the study of Mexican culture. In effect, their priority is generally to learn Spanish and they show less interest in learning Mexican culture. If the student is to be motivated to learn the language as a result of his or her interest in Mexican culture, the system must present the latter in an engaging way.

After carrying out the relevant communicational analysis as suggested by Friedmann [3], the strategy selected was the use of narrative. Telling a story has the virtue of motivating the user to know what is going to happen next, and therefore maintains his or her interest in the application.

On the basis of this strategy the following concept was proposed:

Tell a story through several stages in which the main character is a foreigner visiting Mexico City with whom students can identify and feel empathy towards. The inciting element of the story is the search for the Coatlicue, an Aztec monolith, in different locations of Mexico City. The story must include elements of mystery and plots to keep the interest of the user and incite him to continue. To transmit the information, a series of virtual resources, which young people are used to using, will be offered, such as a mobile phone to communicate with experts, a camera, a GPS, a

music player, interactive books, and a Facebook link to communicate with other students. The credibility of the information is established by the creation of a realistic context.

This concept specifies the manner in which the system will respond to questions such as: how to attract and maintain interest, and how to inform and involve the user. With this concept as a basis, the model of application is established, using the domain specific language defined in the MIIS metamodel. In Fig. 2, we show some of the most important features of "Pierre y la Coatlicue" model.
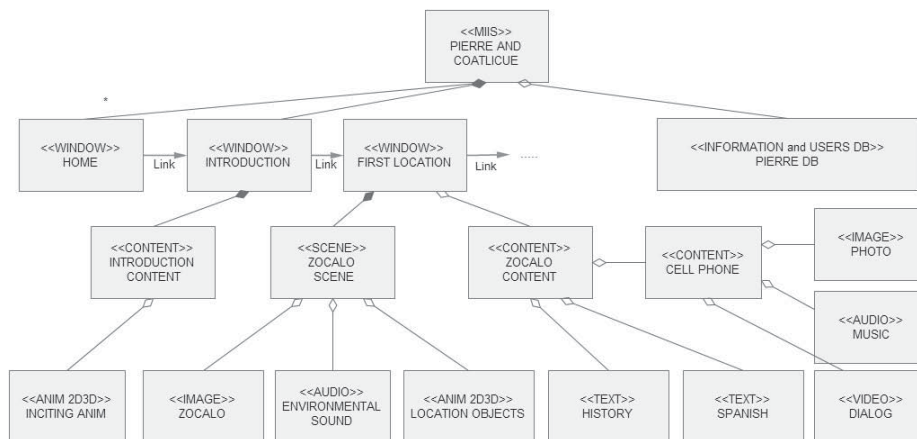
Fig. 2. Case Study Model

5.2   Requirements analysis

In this model, some of the quality attributes are attached to classes, inherited from MIIS and are used to analyze requirements of the system. For instance, to make the system interesting, it is necessary to decide which components of the MIIS will ensure the most interest from the user.

In the "Pierre y la Coatlicue" MIIS, for example, as the story will be told through dialogs and cell phone messages, those components should maintain the users' interest. They must also be informative and credible. In the same way, the inciting element that introduces the story should have same attributes of interest, information and credibility.

As an example, in table 2, some non-functional requirements of the initial animation are described. The analysis is carried out through the attributes of communication and, in parallel, when necessary, the evaluation of the impact of the attribute on the user are established.

Table 2. Proposition for resolution and evaluation of quality goals for the inciting animation

Quality Characteristics	Proposal	Evaluation
Interesting	• Solve a mystery: discover who Coatlicue was, why was she buried and by whom and where is she today	• Time spent viewing animation • Degree of interest to go forward • Inciting incident understanding
Credible	• Characters and background should be in a realistic style • Ambient sounds • Students should indentify with main character	Questionnaire: • Empathy with main character • Who main character is? • Where is action located?
Informative	• The mystery should be linked to Mexican history	Questionnaire: • Historical context understanding

6 Conclusions

In this paper, a methodology that allows the design of an MIIS, taking into consideration its communicational aspects, has being introduced. In order to elicit the requirements of a MIIS without the benefit of the users' input, a MIIS metamodel has being formulated and the model of quality in use of ISO/IEC 25010 has being complemented with a series of new quality attributes of communication. It was shown through a case study that it is possible to establish a system's requirements by analyzing each element of the model through attributes of quality.

From the results of this work, we conclude that communication quality attributes should be part of a quality in use model. They should also be used as elements of analysis to ensure that the system responds to users expectations; it is therefore possible to evaluate their impact on them.

This work opens new perspectives in taking communication problems into consideration while designing MIIS. The formalization of a methodology to design a MIIS is indispensable, given that it offers a multidisciplinary development team communication tools between professionals who come from very different fields. These instruments allow the vision of the work being designed to be unified, and allows the effective drafting of the documents that guide the developers.

References

1. ISO/IEC 25010:2011, Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models (2011)

2. Jones S. and Britton C., Early Elicitation and Definition of Requirements for an Interactive Multimedia Information System, Proceedings of ICRE '96, IEEE, pp.12-19 (1996)

3. Friedmann Anthony, *Writing for visual Media*, Focal Press (2006)

4. Bolchini Davide, Garzotto Franca, Paolini Paolo, Value-driven Design for "Infosuasive" Web Applications, WWW 2008, April 21-25, Beijing, China, pp.745-754, (2008)

5. Cooper Alan, Reimann Robert and Cronin, David, *About Face 3,* Wilwy Publishing, (2007)

6. Callele, Davide, Neufeld, Eric, Schneider, Kevin, Emotional requirements, IEEE Software, (January/February 2008)

7. Zowghi, D., Coulin, C., Requirements Elicitation: A Survey of Techniques, Approaches, and Tools, in Engineering and Managing Software Requirements, edited by Aybuke Aurum and Claes Wohlin, Springer: USA (2005)

8. Coulin, Chad, Sahraoui, Abd-El-Kader, A Meta-Model Based Guided Approach to Collaborative Requirements Elicitation, SE-081010, (28 October 2008)

9. Navarro, Elena et al., A Metamodeling Approach for Requirements Specification, Journal of Computer Information Systems, 47(5): 67-77 (2011)

10. Deissenboeck, Florian et al, Software Quality Models: Purposes, Usage Scenarios and Requirements, *WoSQ'09*, Vancouver, Canada, May 16, 2009

11. Guéneau, Gregory, *Conduite de projets en création numérique*, Eyrolles (2005)

12. Sundar, Shyam, Xu Qian, Bellur Saraswathi, Designing Interactivity in Media Interfaces: A Communications Perspective, CHI 2010: Perspectives on Design, *CHI 2010,* April 10–15, 2010, Atlanta, Georgia, USA.

13. Berki, Eleni et al., Requirements Engineering and Process Modelling in Software Quality Management-Towards a Generic Process Metamodel, Software Quality Journal, 12, 265-283, Kluwer Academic Publishers, Netherlands, 2004

14. ISO/IEC 25012:2008 Software engineering -- Software product Quality Requirements and Evaluation (SQuaRE) -- Data quality model (2008)

Research Preview: Using Improvisational Theatre to Invent and Represent Scenarios for Designing Innovative Systems

Martin Mahaux[1], Anne Hoffmann[2]

[1] PReCISE Research Centre, University of Namur, Belgium,
Martin.Mahaux@fundp.ac.be
[2] Software Engineering Institute, Rijksuniversiteit Groningen, The Netherlands,
A.Hoffmann@rug.nl

Abstract. **[Context and Motivation]** Scenarios are a well-known tool in Systems Design. In particular, they are recognised as an effective means for communicating requirements between business stakeholders and system developers. When designing innovative socio-technical systems, describing creative user experiences is probably one of the first steps. **[Problem]** However, the question of how good scenarios are invented has not been widely discussed in Requirements Engineering. We can also wonder if the written and/or drawn form is the most appropriate for documenting stories. **[Principal ideas]** Building on works inside and outside Requirements Engineering, we suggest that improvisational theatre can be an effective way to invent user experiences in a collaborative way, and to have them instantly documented. **[Contributions]** During a workshop, we showcased one possible form of doing this and discussed it with the audience. We relate this experience here, mention some observations and present our research agenda in this direction.

1. Introduction

We know that scenarios are one of the best techniques for discovering requirements [1]. We also know that, in many situations, requirements are more the result of a collaborative creative effort than they are gathered or translated from the users [2]. The goal we set when using Improvisational Theatre (improv) during this collaborative requirements session is the efficient and effective generation and communication of creative scenarios. In doing this we follow recent advice in Design to focus scenarios on user experience instead of on the system itself [3]. We also introduce fun and play in a rapid prototyping process, as suggested by Schrage for example [4]. In this short paper, we briefly present some related work and describe the 30 minutes demo showcased during the 2nd International Workshop on Creativity in Requirements Engineering (CREARE'12). The video extracted during this presentation is available online from the first author's blog [5]. We finally make some observations and give an insight of our plans for future work.

2. Related Work

We are not the first in thinking about generating scenarios in Requirements Engineering (RE). The most important body of work in this direction probably lies around the CREWS-SAVRE method from Maiden and colleagues [6]. They have developed and experimented a software tool (desktop, then mobile; text-based, then multi-media) to generate scenarios from a use case, based on heuristics for systematic alternate course identification. However, this approach requires a pre-existing scenario or use case. Our technique is more concerned with creating those initial scenarios efficiently. CREWS-SAVRE is also heavily engineered, while our technique would be more natural, and lightweight. More recently, Atasoy [7] has been incorporating techniques from film and sequential art into a tool to provide design teams with an experiential approach towards designing interactive products. Our technique is more dynamic and lightweight, with a focus on the creativity and agility that improv can provide. Both techniques are likely to be complimentary with ours.

Creativity in general has received significant attention in RE. A recent review of these works can be found in [8]. Many creativity techniques exist, but few have been applied to RE. In a prior work [9], [10] we and others have suggested that exercises from the improv world could be used as a training to help RE teams to be more creative in teams. In related domains, many authors have suggested the use of drama, role-play or storytelling in various ways for different purposes, including requirements gathering [11–17]. They indicate many possible ways to use theatre-related concepts in design, with various degrees of success, but lacking scientific validation of empirical results. We build upon this work to suggest the course of action described in the next section.

3. Using Improv as an Experience-Centred Participative Design Technique

3.1 Improv for inventing and representing scenarios

Improvisational theatre, or improv for short, is when actors simultaneously write, direct and play theatre in front of an audience. Actors build on and act out ideas to interpret a theme given to them in real time. Each is unaware of what the other is thinking but acts as if in the same world, imagining what others are doing, seeing and hearing. Each responds to the other actors with new propositions that take the show forward, no matter how bizarre the direction might seem. These propositions build the performance piece by piece. While this discipline exists since the ancient Greeks times, we are interested in its recent modernization as described by Johnstone [18] or Spolin [19]. Along with many followers, they have provided a comprehensive body of knowledge that can be used to invent and represent scenarios on the fly, in a collaborative way.

The expected strengths of using improv as a design technique in RE are the following:

o Improv supports **collaborative creativity**: improv as we use it can be seen as mechanism to create novel, unexpected stories from diverging raw material, and adapts well to stakeholders groups. Interaction between players, and between the audience and the players, is key to improv.

o Improv is **quick and cheap**: given the cost of N people locked in a meeting room, improv's immediacy makes it a very cheap tool compared to other slower techniques.

o Improv is **flexible**: the lack of fixed recording media makes it for a total flexibility, while video recording and a-posteriori editing remains possible.

o Improv is **intuition-based**: improv taps into your intuition to build stories. This neglected idea source complements well with more rational moments in your creative process.

o Improv is **experience-centred**: the focus is not on the designed product, but on the user experience around it.

o Improv enables a **high degree of representation**: actors playing can say much more than a UML diagram or a list of actions in a process diagram, or a drawn storyboard. Emotions in particular are naturally represented.

3.2 The Demo Session at CREARE

To demonstrate our technique at the CREARE workshop, we asked the workshop audience to imagine they were the stakeholders of a project dedicated to build a software application to enhance car-sharing between people. We initially asked for three minutes of informal brainstorming on possible personas and mobility situations. The result was documented on a flip chart, so that they could easily look at it throughout the exercise. The brainstorming gave rise to two very basic personas (a business woman and a conference participant from a foreign country) and some indications on a situation (car-sharing to a conference in Paris in winter). We then immediately started a 5 minute improvised play using the given input. Both of us are trained and experienced improvisers and one of us has worked on a car-sharing project. A third character from the audience who was untrained in improv entered towards the end of the play. During the play, we asked people to note down bits of experience they liked, or disliked, and to derive desired or undesired functionality for our new product. After the play, the audience shared the notes they had made during the improv, and this discussion prompted the generation of further ideas, in conjunction with the unused ideas from the initial brainstorming.

3.3 Observations from the demo session

Despite its limited duration, our improvisation at CREARE workshop resulted in the generation of numerous creative and interesting ideas. It was interesting to see how the audience was continuing the story collaboratively during that discussion, inventing alternative courses. Some participants seemed to become even more creative throughout the discussion of further possibilities. It is clear that a novel and useful scenario had been invented and represented for the stakeholders in an effective

way. "Unrealistic" and "crazy" behaviors during the play were not seen as irritating, but rather served to release the audience members from constrained thinking and thereby triggered more novel ideas. Overall we received good feedback, and the audience had both enjoyed themselves and generated a significant number of requirements and solution concepts for the car-sharing domain.

4. Future Work

The above described session can however certainly not be considered as a validation of our technique. It is in our plans to do this validation work in a near future. This will on one hand let us assess the strong and weak points of using improv compared to other techniques, and on the other hand refine our techniques and prepare guidance on when and how to use improv, which improv form to use in which circumstances, how to facilitate improv sessions, how to document them and how to train people.

Empirically assessing the efficiency and effectiveness of such a technique will however not be easy, for two main reasons. First, representative measures will be hard to define, and to observe. To mitigate this, we have done some preliminary work to understand creativity, and in particular collaborative creativity, in RE [8], [20]. This work should help us define measures, however imperfect, in this "soft" domain. Second, there are many potential complex variables that have an influence on the final results, including the desired kind of creativity (in [8], we show there are many different kinds of creativity), the level of details of desired requirements, the freedom in the scope, the product type (custom or market), the place in time within the process, the training or team building previously received, the available time, the organizational climate, the experience of the facilitator... This requires doing as many experiments as possible and carefully recording the state of these variables during experiments, keeping some of them fixed.

5. Acknowledgments

6. References

[1] K. Weidenhaupt, K. Pohl, M. Jarke, and P. Haumer, 'Scenarios in system development: current practice', *IEEE Software*, vol. 15, no. 2, pp. 34–45, Apr. 1998.

[2] N. Maiden and A. Gizikis, 'Where do requirements come from?', *Software, IEEE*, vol. 18, no. 5, pp. 10–12, 2002.

[3] M. Hassenzahl, 'Experience Design: Technology for All the Right Reasons', *Synthesis Lectures on Human-Centered Informatics*, vol. 3, pp. 1–95, Jan. 2010.

[4] M. Schrage, *Serious play : how the world's best companies simulate to innovate*. Boston mass.: Harvard Business School Press, 2000.

[5] M. Mahaux, 'Creativity, Collaboration, Sustainability | Martin's Blog'. [Online]. Available: http://info.fundp.ac.be/~mma/wordpress/. [Accessed: 04-Apr-2012].

[6] N. A. . Maiden, S. Minocha, K. Manning, and M. Ryan, 'CREWS-SAVRE: systematic scenario generation and use', in *1998 Third International Conference on Requirements Engineering, 1998. Proceedings*, 1998, pp. 148–155.

[7] B. Atasoy and J.-B. Martens, 'STORIFY: a tool to assist design teams in envisioning and discussing user experience', in *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems*, New York, NY, USA, 2011, pp. 2263–2268.

[8] M. Mahaux, A. Mavin, and P. Heymans, 'Choose Your Creativity: Why creativity means different things to different people', in *Proc. REFSQ'12, Essen, Germany, 2012.*

[9] M. Mahaux and N. Maiden, 'Theater Improvisers Know the Requirements Game', *IEEE software*, vol. 25, no. 5, pp. 68–69, 2008.

[10] A. Hoffmann, 'REIM - An Improvisation Workshop Format to Train Soft Skill Awareness', in *To appear in Proceedings of CHASE'12*, Zurich, Switzerland, 2012.

[11] M. Arvola and H. Artman, 'Interaction walkthroughs and improvised role play', *Design and semantics of form and movement*, p. 42, 2006.

[12] L. Gongora, 'Exploring creative process via improvisation and the design method RePlay', in *Proceedings of the 1st DESIRE Network Conference on Creativity and Innovation in Design*, Lancaster, UK, UK, 2010, pp. 44–51.

[13] I. D. Sorby, L. Melby, and G. Seland, 'Using scenarios and drama improvisation for identifying and analysing requirements for mobile electronic patient records', *Requirements Engineering for Sociotechnical Systems*, pp. 266–283, 2005.

[14] S. Boess, 'Making role playing work in design', *Design and semantics of form and movement*, p. 117, 2006.

[15] E. Brandt and C. Grunnet, 'Evoking the future: Drama and props in user centered design', in *Proceedings of Participatory Design Conference (PDC 2000)*, 2000, pp. 11–20.

[16] F. Marquis-Faulkes, S. J. McKenna, P. Gregor, and A. F. Newell, 'Scenario-based drama as a tool for investigating user requirements with application to home monitoring for elderly people', *HCI International, Crete, Greece*, 2003.

[17] N. Boulila, A. Hoffman, and A. Hermann, 'Using Storytelling to Record Requirements: Elements for an Effective Requirements Elicitation Approach', in *Proc. MERE'11*, Trento, Italy, 2011.

[18] K. Johnstone, *Impro: Improvisation and the Theatre*. Routledge, 1981.

[19] V. Spolin, *Improvisation for the Theater 3E: A Handbook of Teaching and Directing Techniques*, 3rd ed. Northwestern University Press, 1999.

[20] M. Mahaux, O. Gotel, K. Schmid, A. Mavin, L. Nguyen, M. Luisa, and G. Regev, 'Factors Influencing Collaborative Creativity in Requirements Engineering: Analysis and Practical Advice.', submitted to *RE'12.*

5 Requirements Prioritization for Customer Oriented Software Development (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'12

Georg Herzwurm[1], Wolfram Pietsch[2]

[1] Department for Business Administration and Information Systems, esp. Business Software,
Universität Stuttgart, Keplerstr. 17, 70174 Stuttgart, Germany
`herzwurm@wi.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'12 represents the 3rd Workshop on Requirements Prioritization for customer oriented Software Development (RePriCo'12) held at the 18th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ2012).

The workshop served as a platform for the presentation and discussion of new and innovative approaches to prioritization issues for requirements engineering with a focus on customer orientation.

As far as prioritization is an essential task within requirements engineering in order to cope with complexity and to establish focus properly two perspectives can be identified:

- From a formal standpoint of view prioritization is merely a matter of choice of the right specification method and granularity of analysis.
- From a practical perspective it is a matter of customer orientation also: consensus must be achieved about the appropriateness of requirements from the view of the customers and fed back into the process.

From our point of view customer orientation means a strategy for the selection of action alternatives, which gives the target "satisfaction of customer needs" the highest preference. Therefore requirements prioritization methods and approaches are not limited to bespoke software but affect standard software also.

We are glad about holding RePriCo'12 for the third 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.

RePriCo'12 attracted 9 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, three submissions were accepted as full research papers and two submissions as short papers.

The submissions comprise current research findings from various fields: prioritization of user experience requirements to support practitioners (Pariya Kashfi, Agneta Nilsson, Robert Feldt); prioritization of requirements by using requirements relations (Constanze Kolbe); discussion of consensus in requirements engineering from the viewpoint of discourse ethics (Alexander Rachmann); tool-supported requirements prioritization and negotiation in consideration of dynamics and interactivity (Andreas Reiser, Benedikt Krams, Mareike Schoop); tackling prioritization in business-process-driven software development (Norman Riegel, Joerg Doerr, Oliver Hummel).

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 2010 as well as 2011 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 Samuel Fricker as REFSQ2012 workshop chair and Vanessa Stricker as REFSQ2012 organizational chair for their professional support. Last but not least we thank Sixten Schockert and Benedikt Krams for their effort in organizing RePriCo'12. We are confident in hosting RePriCo in 2013 once more and are looking forward to welcoming many participants again.

2   Organization

2.1   Program Committee

Chair.
Prof. Dr. Georg Herzwurm, Universität 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, Software Management Consultant, Germany
Priv. Doz. Dr. Andrea Herrmann, Infoman AG, 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
Dipl.-Wirt.-Inf. Sixten Schockert, Universität Stuttgart, Germany
Prof. Dr. Klaus Schmid, University Hildesheim, 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, Universität Stuttgart, Germany
Dipl.-Wirt.-Inf. Sixten Schockert, Universität Stuttgart, Germany

Supporting Practitioners in
Prioritizing User Experience Requirements

Pariya Kashfi, Agneta Nilsson, Robert Feldt

Software Engineering Division
Department of Computer Science and Engineering
Chalmers University of Technology and Gothenburg University
`pariya.kashfi,agnnil,robert.feldt@chalmers.se`

Abstract. The success of, in particular, market-driven and customer-oriented software systems is dependent on finding a proper balance among various quality requirements. There is a gap in current theory and practice in prioritizing quality requirements, especially those quality requirements that are not related to performing a task or accomplishing a goal such as joy. To bridge this gap, a shared understanding of these types of requirements is required. This paper includes a review of the current theories, i.e. quality models in software engineering, and user experience models in interaction design. We then present our results from comparing models from each field. We conclude that the models are complementary, and can and should be merged to form a combined model. The model will bring insight into prioritization by introducing various aspects of user experience, its composing elements, and their functional relation.

1 Introduction

Researchers have emphasized prioritization of Quality Requirements (QR) in software (SW) development. Nevertheless, guidelines and methods in gathering, prioritizing, and documenting QRs are limited[1–5]. Especially for those QRs that are not related to performing a task or accomplishing a goal, such as emotional connection, joy, curiosity, and excitement. We refer to them as *non-task-related* or *non-instrumental* requirements, as opposed to *task-related* or *instrumental* requirements. The history of studying non-task-related requirements, or in a more general term *User eXperience* (UX), goes back to the 90's [6–9]. So far, within Software Engineering (SE), few studies have taken UX into account. In fact, it can be argued that understanding and controlling UX is a missing component in the current QR literature.

To properly include and prioritize UX requirements in SW development, it is necessary for practitioners to understand UX and its composing elements. Practitioners need to have knowledge on how to manage UX requirements, discover effects of other requirements on UX, and accordingly prioritize the requirements. To provide a white-box view and a shared understanding of UX, the first step in our research is to study the existing models and investigate their potential to support practitioners in prioritizing requirements. This paper reviews related theories in SE, and Interaction Design (ID), the latter being the leading field for UX research (Sects. 2, 3). It identifies relevant SW qual-

ity models, and UX frameworks[1], selects two, and compares and contrasts them from practitioners' perspective (Sect. 3). Finally, it discusses the implication of our findings for requirements prioritization, and future research (Sect. 4).

2 Related Work

Ultimate success of SW as well as business goals, such as the market share, profit and company image, will be reached by satisfying various user needs [10]. Task-related and non-task-related user needs have been in focus for more than two decades in ID [6], but not received much attention in SE. Requirements Engineering (RE) have evolved and addresses part of task-related requirements but QRs, especially non-task-related QRs, are still insufficiently supported [1, 3]. In particular, while usability is only one aspect of UX, it has been discussed to be one of the QRs that are challenging to specify[3]. This difficulty increases when dealing with other aspects of UX than usability. The challenge in prioritizing UX requirements is that UX is related to various functional, and QRs, e.g. a certain interface performance requirement might not be motivated by its effect on the task at hand but on the frustration its non-fulfillment would have on the user.

Hassenzahl et al. [11] define UX as "a consequence of a user's internal state (pre-dispositions, expectations, needs, motivation, mood, etc.), the characteristics of the designed system (e.g. complexity, purpose, usability, functionality, etc.) and the context (or the environment) within which the interaction occurs (e.g. organizational/social setting, meaningfulness of the activity, voluntariness of use, etc.)." A related concept in SE is *Quality in Use* (QiU) that according to ISO25010 [12] is "the degree to which a product or system can be used by specific users to meet their needs to achieve specific goals with effectiveness, efficiency, freedom from risk and satisfaction in specific contexts of use." ISO25010 also defines *Product Quality* (PQ). PQ relates to static properties of software and dynamic properties of the computer system and their influences on QiU. ISO25010 is an extension to ISO9126 [13] that included *Internal* and *External* Quality (I&EQ) models (combined in ISO25010 to form PQ), and QiU.

Even though QiU model does not include any definition, or direct reference to UX, it has been applied in the context of UX by SE researchers. For instance, Doerr et al. [14] tried to discover the relationship between I&EQ and UX that in their view was equivalent to QiU in ISO9126. From a prioritization perspective, such studies can bring knowledge on how different requirements affect each other, and result in more informed decisions in RE. In a later study in 2007 [15], Doerr et al. suggested using a questionnaire in RE, to prioritize the product features and improve the product's UX. In addition, they proposed a SW quality model named AMUSE (Appraisal and Measurement of User Satisfaction). Prioritizing UX requirements and considering the effect of other requirements on UX is to some extent covered in AMUSE study. In 2006, the FUN project started with focus on pattern-based approaches in developing SW with positive UX [16]. FUN is based on a quality model called e4FUN [17]. One of the results of FUN is KREA-FUN workshop [17], a systematic approach to improve the joy-of-use. The method helps in eliciting UX related requirements, but also prioritization. Among

---

[1] The terms "model" and "framework" are used interchangeably in this paper.

other steps, the method includes evaluating and rating various requirements. There are other studies related to UX in SE including [18–22]. Although these studies do not touch prioritization, they cover various concepts related to UX such as emotional requirements, user motivations, user values, user stories, and so on.

In summary, there have been few research efforts related to UX in RE in general, and prioritization in particular. For a background to prioritization of quality requirements, we refer to [1]. In a recent study [2] practitioners listed usability as a key QRs but used no or ad hoc methods in prioritization of QRs. To our knowledge, prioritization literature does not go beyond usability [1–4], and non-task-related aspects of UX remain untouched. Finally, since there is little support even for the definition and description of UX requirements, it has never been directly addressed in prioritization methods.

3 Preliminary Results

As the initial step in providing a shared understanding of UX for practitioners, this section reviews a number of UX frameworks, and compares and contrasts two models one from SE and ID respectively to identify their strength and weaknesses from practitioners' perspective.

First, it is important to specify a definition of UX that is understandable for practitioners. Even in ID, there is not yet a widely accepted definition of UX [6, 23]. After reviewing the current UX definitions, we chose the definition by Hassenzahl and Tractinsky [11] (see Sect. 1). The definition includes the influencing factors on UX, i.e. user, product and context, and provides examples of each factor, which we find supports a better understanding.

Some frameworks view UX as a holistic concept that cannot be divided into objective elements. Others try to find objective elements that influence UX, hence make it possible to consider UX in designing SW. Among the existing frameworks, the most clear and comprehensive ones in our view are as follows.

Hassenzahl's framework [7] is based on his categorization of product attributes, i.e *pragmatic* and *hedonic*. Hassenzahl defines hedonic quality as "a quality aspect that addresses human needs for social power, novelty and change" [23]. On the other hand, pragmatic aspects are related to instrumental qualities of a product [7]. This framework includes both designer's, and user's perspectives.

Zimmermann's framework [6] is an integration and extension of other frameworks, in particular Hassenzahl's. The framework includes three phases of experience, *sensory encounter*, *interaction phase* and *evaluation phase*. In contrast to Hassenzahl that considers both designer's and user's perspectives, Zimmermann only focuses on the user's perspective. Also, while Zimmermann indicates the role of product features in UX, the model seems to be missing that aspect.

Mahlke's framework, hereafter UXF, has four composing blocks (i) theoretical considerations (ii) methodological contributions (iii) empirical studies (iv) recommendations for the development of interactive systems. UXF includes Hassenzahl's hedonic and pragmatic quality aspects, and considers the influence of product features, use, and context characteristics on UX. UXF includes three UX components (i) perception of instrumental qualities (ii) perception of non-instrumental qualities (iii) emotional user

reactions, and their interrelation. One strength of UXF compared to Hassenzahl's is the division of the categories of instrumental, non-instrumental and emotion-related quality aspects into sub-dimensions which makes measurement of those aspects easier. Considering its strengths, we have chosen UXF for our research.

(a) Identified overlaps   (b) Our research process
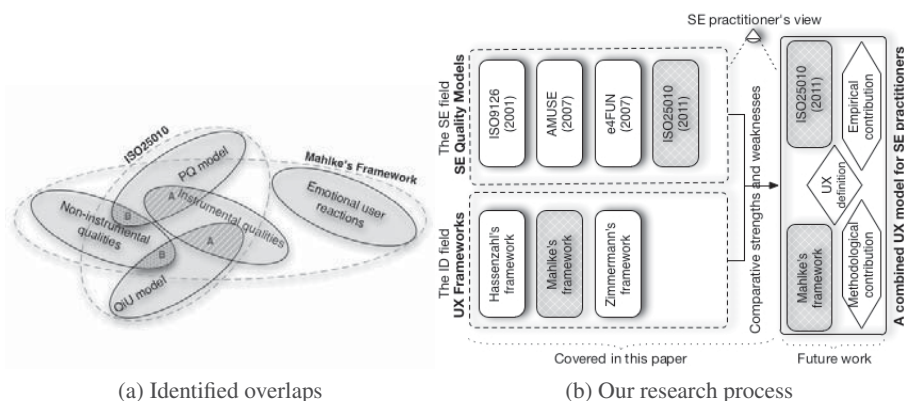
Fig. 1: Overlaps in the models on a conceptual level; and the research process

Comparison to Quality models

For providing a shared understanding of UX for practioners, we compare ISO25010 and UXF. We have chosen ISO25010 for this comparison because it is the latest model presented, has the status of a standard, and often cited as important by practitioners. We drew the comparison with a focus on the questions (i) which types of SW qualities are included? (ii) which perspectives in analyzing UX are considered? (iii) which components, or influencing factors are defined for UX?

ISO25010 focuses on defining and evaluating quality requirements of SW, while UXF focuses on design for positive UX, and evaluating the design. UXF provides a clear distinction between instrumental, and non-instrumental qualities, while in ISO25010, PQ and QiU have overlaps in this regard. In Mahlke's view, what eventually influences UX is how a user perceives different qualities of the SW. In contrast, ISO25010 considers only the developer's point of view and does not include user's point of view. On the other hand, the focus on user's view has made UXF more difficult to understand for practitioners. The two models have the same definitions, or consider the same composing elements for some concepts such as usability, learnability and efficiency. This indicates an overlap as depicted in Fig. 1a. The area marked A shows an overlap between instrumental qualities in UXF and the models in ISO25010. A is bigger than B since we found more similarities between these models in instrumental compared to non-instrumental aspect of qualities. The comparison is summarized in Table 1.

Aspect	ISO25010	UXF
Instrumental qualities	strong	weak
Non-instrumental qualities	weak	strong
Emotional user reactions	missing	strong
Influences on interaction	weak	strong
User's perception	missing	strong
Designer's perspective	weak	weak
UX consequences	missing	strong
Methodological contribution	missing	weak
Empirical contribution	missing	weak

Table 1: Comparison between ISO25010 and Mahlke's framework.

4   Discussion and Conclusion

Our findings imply that there is limited support to deal with non-task-related requirements, and understanding of UX is generally shallow in SE. In SE, UX is treated the same as other QRs. While providing an opportunity to benefit from the existing advice on measuring various quality characteristics in SE, and to some extent influence UX during SW development, this leads to a narrow view on UX since not all aspects of UX have been covered in the existing quality models. This makes it harder not only to prioritize UX requirements but also to discuss and trade-off requirements in general considering the dependency of UX to different functional and QRs.

Hence, there is a need for a framework that supports these issues, and makes them tangible and understandable for practitioners. Based on our study, such a framework should include various aspects of UX, definitions of key UX elements, and their functional relations. Therefore the framework will support prioritizing among UX requirements as well as in relation to other requirements.

Future plans are to develop a framework as depicted in Fig. 1b. We believe any support provided for practitioners should have a strong theoretical basis, and we find the existing theories complementary. The weaknesses in the current models can be overcome by merging them. For instance, the new framework should consider both developer's, and user's perspectives, it should provide details on non-instrumental, and emotion-related qualities, and consider factors influencing UX, and the consequences of UX on user's decisions and interactions. Also, there is a need to find the barriers in current RE practices, and provide guidelines and methods for how to improve them. Only then can we support prioritization of not only UX requirements but develop prioritization methods that cover the whole range of requirements required for long-term success.

References

1. R. B. Svensson, M. Host, B. Regnell, Managing Quality Requirements: A Systematic Review, in: 2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications, Ieee, 2010, pp. 261–268.

2. R. B. Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, R. Feldt, A. Aurum, Prioritization of quality requirements: State of practice in eleven companies, in: 19th International Requirements Engineering Conference, IEEE, pp. 69–78.

3. L. Chung, J. do Prado Leite, On Non-Functional Requirements in Software Engineering, Vol. 5600 of LNCS, Springer Berlin / Heidelberg, 2009, pp. 363–379.

4. R. Berntsson Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, R. Feldt, Quality Requirements in Industrial Practice -An Extended Interview Study at Eleven Companies, IEEE Transactions on Software Engineering (2011) 1–14.

5. M. Glinz, On Non-Functional Requirements, in: 15th IEEE International Requirements Engineering Conference (RE 2007), IEEE, 2007, pp. 21–26.

6. P. G. Zimmermann, Beyond Usability–Measuring Aspects of User Experience, Ph.D. thesis (2008).

7. M. Hassenzahl, The thing and I: understanding the relationship between user and product, Funology: from usability to enjoyment (2003) 31–42.

8. P. W. Jordan, Designing Pleasurable Products: An Introduction to New Human Factors, Taylor & Francis, 2000.

9. S. Mahlke, User experience of interaction with technical systems, Ph.D. thesis, Berlin, Technical university (2008).

10. D. Kerkow, Don 't have to know what it is like to be a bat to build a radar reflector-Functionalism in UX, in: E. Law, A. P. Vermeeren, M. Hassenzahl, M. Blythe (Eds.), Towards a UX manifesto, COST294–MAUSE affiliated workshop, 2007, pp. 19–25.

11. M. Hassenzahl, N. Tractinsky, User experience – a research agenda, Behaviour & Information Technology 25 (2) (2006) 91–97.

12. ISO25010, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models, Vol. 2011, International Organization for Standardization, Geneva, Swiss, 2011.

13. ISO9126, Software engineering - Product quality, Tech. rep., International Organization for Standardization, Geneva, Swiss (2001).

14. J. Doerr, D. Kerkow, Total control of User Experience in Software Development – a Software Engineering dream?, in: E. Law, E. Hvannberg, M. Hassenzahl (Eds.), Proceedings of the The Second COST294MAUSE International Open Workshop User ExperienceTowards a Unified View, 2006, pp. 94–99.

15. J. Doerr, S. Hartkopf, D. Kerkow, D. Landmann, P. Amthor, Built–in User Satisfaction – Feature Appraisal and Prioritization with AMUSE, 15th IEEE International Requirements Engineering Conference (RE 2007) (2007) 101–110.

16. FUN project (2006). URL http://fun-of-use.org/

17. D. Kerkow, C. Graf, KREA-FUN : Systematic Creativity for Enjoyable Software Applications, in: FUN 2007 Proceedings: Workshop for Design Principles for Software That Engages Its Users and Facing Emotions: Responsible Experimental Design, 2007.

18. D. Callele, E. Neufeld, K. Schneider, An Introduction to Experience Requirements, 2010 18th IEEE International Requirements Engineering Conference (2010) 395–396.

19. D. Callele, E. Neufeld, K. Schneider, Emotional Requirements, IEEE Software 25 (1) (2008) 43–45.

20. N. Maiden, Requirements and Aesthetics, IEEE Software 28 (3) (2011) 20–21.

21. A. Sutcliffe, Emotional requirements engineering, in: 2011 IEEE 19th International Requirements Engineering Conference, IEEE, 2011, pp. 321–322.

22. D. Bolchini, J. Mylopoulos, From Task-Oriented to Goal-Oriented Web Requirements Analysis, in: Proceedings of the Fourth International Conference on Web Information Systems Engineering, WISE '03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 166—-.

23. M. Hassenzahl, The Effect of Perceived Hedonic Quality on Product Appealingness, International Journal 13 (4) (2001) 481–499.

Requirements Prioritization by Using Requirements Relations

Constanze Kolbe[1]

[1]RIF e.V.
Joseph-von-Fraunhofer Str. 20,
44227 Dortmund, Germany
`constanze.kolbe@rif-ev.de`

Abstract. During the planning process a customer utters a variety of requirements on a product and specifies them. Between the super-ordinated and their detailing requirements vertical relations do occur, which leads to the establishment of a requirements hierarchy. At the same time relations (e.g. conflicting, supporting relations) do occur between these requirements on each hierarchy level.

During the prioritization of requirements, requirements relations must be taken into account. Horizontal relations enable on the one hand to deliberate about whether an absolute weighting will be sufficient or weather a relative weighting of requirements which belong to the same level is necessary to heighten the precision of a weighting. This enables the reduction of weighting effort.

Vertical requirement relations help to identify inconsistencies between the weight of a super-ordinated requirement and the weightings of their detailing requirements. A computer-aided procedure will be presented, that uses requirement relations, in order to support the prioritization of stated requirements.

Keywords: prioritization, requirements, requirements relations

1 Introduction

In the context of the product planning process customer requirements and their weightings have to be captured [1]. By means of priorities requirements can be sorted into a ranking. This enables to handle with resources stringency and to avoid the realization of needless product characteristics. The customer further specifies his stated requirements, so that in each case a new requirements level emerges [2]. Requirements of a new established level also have to be weighted by the customer.

One problem of requirements prioritization is that the whole weighting process can become very work-intensive. The more precisely the weighting of requirements must be, the more complex is the weighting process. In some cases, a high accuracy of weighting of a requirement is not needed, as their influence on the overall result of the planning process is only low. If the weightings of those requirements are nonetheless determined very precisely, an unnecessary effort during the weighting process arises.

At the same time, it should be determined which requirement weightings have to be very precise, because they mostly influence the result of the planning process. Regarding conflicting requirements the requirement weighting is an important aspect during their translation into adequate product characteristics [3], thus a higher accuracy of their weightings is necessary.

Due to weighing errors of the customer, there is a risk that the weight of the superordinated requirement and the weightings of its detailing requirements are not consistent with each other. Possible errors remain undetected, if inconsistencies concerning the vertical direction are not taken into account. This could have a negative impact on the planning process and could lead to a noncompliance of the product characteristics with the imaginations of customer. A possible inconsistency concerning the vertical direction occurs e.g. when an extreme important requirement is concretized with an amount of unimportant requirements. In this case either an incompleteness of detailing requirements is given or there exists an over-estimation of the super-ordinated requirement or an under-estimation of the detailing requirements.

In order to solve the mentioned problems, requirements relations have to be considered during the requirements prioritization. In section 2 different types of requirement relations will be presented. It will be described which ones have to be considered in order to handle the mentioned problems during the weighting process.

2   Types of Requirements Relations

While setting and refining requirements, the requirement model shown in figure 1 is set up. The model will be extended and detailed during the whole development process and is the basis for a structured transferring of requirements into concrete product characteristics [4]. It consists of three dimensions [5]: The first dimension shows the transition of abstract requirements to detailing requirements. The second dimension is the classification of content of requirements. The content-related structuring of requirements into the categories „obligations", „surroundings", „economy", „information", „qualification" and „technical-functional aspects" according to [6] can be used as the classification of contents of the requirements within the requirements model. The third dimension is the completeness and disjunction of requirements and their relations.

According to [2], [5], [7], [8], [9], and [10], existing approaches for structuring of requirement relations concerning their direction distinguish between horizontal and vertical requirement relations (fig. 1).
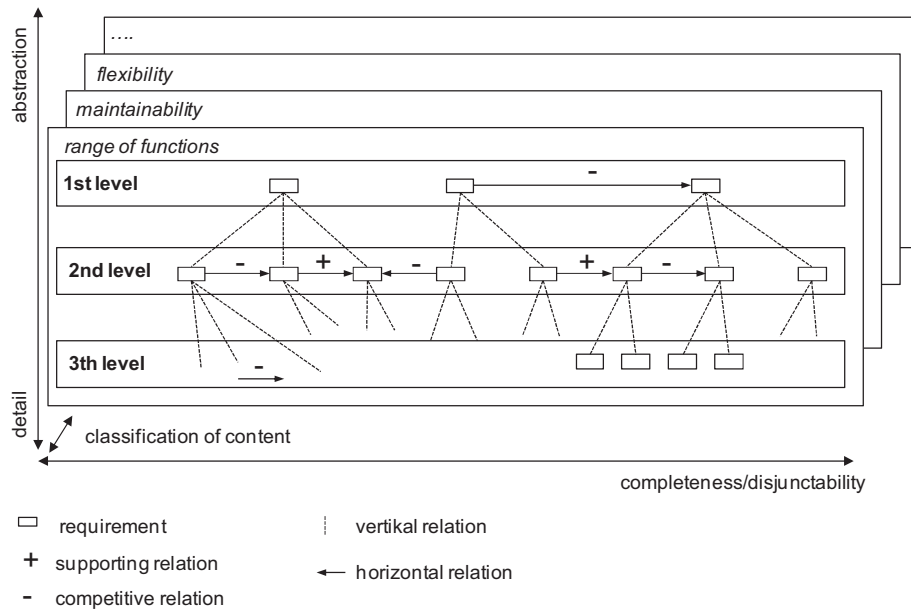
Fig. 1. Dimension of the requirement model [following 5]

In order to receive an overview of the product that has to be developed, at the beginning of a planning process abstract, and hence, imprecise requirements are gathered [11]. When a customer specifies his given requirements by means of detailing requirements, **vertical requirements relations** (concretization relations) do occur between them [2], which leads to the establishment of a requirements hierarchy [2]. In figure 1, the most abstract requirements are located on the 1st level and on the 2nd level the detailing requirements of the requirements belonging to 1st level are located. The requirements of 1st level are substituted by the requirements of 2nd level. Thus, vertical relations give information about which requirement details another requirement and for which requirements a consistency analysis in vertical direction has to be performed. During the consistency analysis it can be ascertained whether the weight of a super-ordinated requirement and the weightings of its detailing requirements of the next level are consistent with each other or whether weighting errors do exist.

Horizontal requirement relations are not caused by concretization of an abstract requirement but arise when requirements being on the same level of abstraction are set in relation to each other [9] and interact by technical and logical dependencies [12], [2]. They occur for example because of technological risks, or if a requirement has a high influence on requirements on the implementation costs of the product characteristics. Together with the requirements hierarchy horizontal relations constitute a requirements model. These horizontal relations become more concrete on each further level and will be substituted by the relations of the next lower level. Horizontal relations can be divided concerning their sort of effect into the following types: "**competitive**" (the fulfillment of a requirement negatively influence the fulfillment of another

requirement) and "**supporting**" (the fulfillment of a requirement supports the fulfillment of another requirement) [10]. For identifying requirement relations the knowledge-based methods for an automatic computer-aided identification of concretization as well as supporting and competitive relations developed in [13], [14], [15] are suitable. Due to these computer-aided methods requirement relations can be automatically identified with low effort.

By means of horizontal relations it can be determined, which part of the requirements has to be weighted with a high precision and for which part a less precise weighting is acceptable. Due to that, a requirement weighting on the horizontal level can be carried out goal-oriented and with reduced effort.

Concerning requirements, which are connected to each other over conflicting relations, often compromises are needed during their realization. Conflicting requirement relations give an indication that the accuracy of the associated requirements weightings should be quite high, as they have a strong influence on the results of the planning process.

In the case a requirement does not have any horizontal or only a supporting horizontal relation to other requirements, it can be assumed that, this requirement does not need to be weighted very precisely. The reason for that is that during the selection of product characteristics in planning process, the identified requirements will not be in competition against each other, what means that between those a compromise must not be reached later.

There already exist different approaches to prioritise requirements of a requirements model. They can mainly be divided into absolute and relative weighting. In the following the approaches will be presented and their advantages and disadvantages will be shown.

3 Absolute and Relative Weighting

The easiest form of requirement weighting is the **absolute weighting** where the customer uses a rating scale (here: 1 to 5) to weight each of his requirements without setting them in relation to one another (Fig. 2) [16].

1	2	3	4	5
unimportant	rather unimportant	important	very important	extremely important
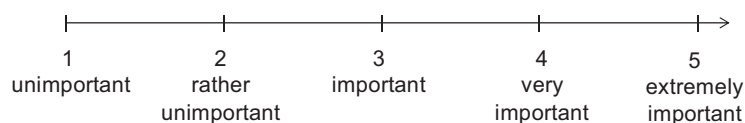
Fig. 2. Absolute weighting using rating scale

The advantage is that the priorities can be established with little effort. Furthermore, the customer needs to define a weighting between "unimportant" (1) and "extremely important" (5) for each requirement (fig. 2), which allows an analysis of consistency concerning the vertical direction within the requirements hierarchy. Thereby, a possible over- or under-estimation of requirements can be detected. The disadvantage of an absolute weighting is that the results are not very precise, because a requirement is not

compared to another requirements during the prioritization. This creates the danger that e.g. the first of two conflicting requirements wrongly receives a lower weighting than a second requirement, although the customer would have prefer the first requirement in the framework of a direct comparison. Due to that, an absolute weighting is only for this part of requirements sufficient, for which a high accuracy of prioritization is not needed, and thus for requirements with a low overall importance as well as for requirements with a supporting or without any relation to other requirements.

A further form of requirement weighting is the **relative weighting** by means of paired comparison in which a requirement is compared with other requirements (2: is more important than; 1: equally important; 0: less important) [17]. Contrary to the absolute weighting, the relative weighting achieves more precise results. The disadvantage of a relative weighting is a fast-growing complexity [17], as each requirement has to be compared with each other requirement. The relative weighting is only worthwhile for requirements, whose accuracy of the weightings must be quite high as it is the case for requirements which are not of an overall unimportance and with a conflicting relation to other requirements.

A method that uses relative weightings in order to prioritize a requirements hierarchy is named analytic hierarchy process (AHP). Using this method, requirements of the same level, which do refine together the same super-ordinated-requirement of the next superior level, are relatively weighted in a paired comparison [18]. With it, requirements, which do refine different super-ordinated requirements, are not compared to each other during their weighting. In case when these both requirements do compete with each other, the customer is not able to consider their relative weightings to each other, although exactly this information is the most important aspect when dealing with conflicts of requirements realization. One further problem is that relations between requirements belonging to one hierarchy level are not admissible in the framework of AHP.

In contrast to AHP, the prioritization method analytic network process (ANP) considers requirements relations, during the relative weighting of requirements [19]. However, the disadvantage of AHP and ANP is that the above mentioned analysis of consistency concerning the vertical direction cannot be performed. The reason for this is that relative and not absolute weightings are assigned to the requirements of a hierarchy or network. Due to that, the customer is not able to interpret a relative weighted requirement, because a relative weighting does not express if a requirement is unimportant or extremely important, as it is the case with an absolute weighting on a scale from 1 (unimportant) to 5 (extremely important) (fig. 2). Thus, an under-or over-estimation of requirement´s weightings cannot be detected when using relative weightings.

The aim of this research is to develop a procedure for the performance of a requirements weighting, which uses requirements relations, in order to utilize both the advantages of the absolute and the advantages of the relative weighting. By means of these procedure inconsistencies within the vertical direction concerning a possible incompleteness or an over- or under-estimation of requirements should be detected. Furthermore, the procedure has to give the information, when the allocation of absolute weightings (from 1 to 5) to requirements will be sufficient or when the effort of

an additional relative weighting is worth. Therefore the procedure has to consider the information about supporting and conflicting relations between requirements. Goal is an appliance of the procedure with a low workload. To achieve this goal the procedure has to be implemented in the forms of a software component, which enables an automatic execution of the procedure.

4   Using Requirements Relations During the Prioritization Process

In the following, the single steps of the new developed procedure for requirement prioritization (prioritization analysis) will be presented. Thereby, it is shown how the identification of requirement relations described in chapter 2 can be used to support the prioritization. The proceeding will be applied by capturing requirements on a continuous conveyor in manual commissioning of a pharmaceutical trade. These requirements are formulated and weighted by a logistics expert who is an employee of a trade company in Germany.

Within figure 3 the steps of the developed procedure is presented.
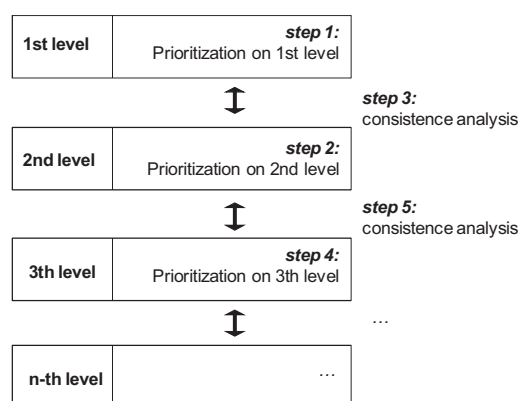
Fig. 3. Steps of prioritization analysis

The requirements on 1st level of the requirements model (figure 1) have to be weighted in the first step. In a second step, the 2nd level of the model will be weighted. Afterwards, in step 3, the weightings of the 1st and 2nd level are set in relation to each other and a consistency analysis concerning the vertical direction will be carried out. During this check a possible incompleteness of detailing requirements and an over- or under-estimation of requirements belonging to both levels can be detected. In step 4, the 3rd level will be weighted and compared with 2nd level in context of a consistency analysis concerning the vertical direction (step 5). So the procedure changes between the requirement weighting on horizontal level and the weighting of requirements connected via vertical relations of two levels (figure 3) till the very bottom level of the requirement model is reached.

4.1 Prioritization Analysis in Horizontal Direction

During the prioritization analysis in horizontal direction at first conflicting requirements of one hierarchy level have to be weighted relatively to one another, in order to thus achieve a ranking of those requirements. Subsequently the whole amount of requirements of one level has to be prioritized by means of absolute weightings. The additionally absolute weighting of already relatively weighted requirements is therefore necessary to set them in relation to the remaining not-relatively weighted requirements and to carry out the named consistence analysis in vertical direction.

Steps of the Analysis in Horizontal Direction.

1. Relative Weighting Regarding Conflicting Relations between Requirements.
In figure 1, conflicting requirements are demonstrated on the 1st and 2nd level by using the sign "-". The conflicting requirements identified by the software component are presented to the customer. The customer has to detect those of these requirements which are in his opinion of a low overall importance. These requirements do not need to be assigned with weightings having a high precision, because a trade-off between them does not have a high influence on the customer satisfaction. They must therefore not relative weighted; instead an absolute weighing is sufficient. The remaining conflicting pairs of requirements are weighted relatively by the customer. Thus the conflicting requirements and also partly requirements having no or a supporting relation to other requirements as well as unimportant but conflicting requirements, are brought into a ranking.

2. Absolute Weighting of all Requirements.
The requirements ranking received within step 1 is basis for the following step two. Within this step 2 the customer has to assign absolute weightings to all requirements of one level. In doing so, the absolute weightings are not allowed to be contradicting to the previously established ranking of requirements (step 1).

   For the remaining requirements which cannot be ordered by relative weighting the absolute weightings are intuitively allocated by the customer without comparing them with other requirements. In comparison to the direct absolute weighting (without paired comparison), the advantage of this approach is that a ranking of conflicting requirements has been established by means of paired comparison before assigning absolute weights to those requirements (step 2). Additionally, the gained information about the relative weightings can be considered during the planning process as soon as a decision for or against product characteristic has to be taken, for which the relative weighting between conflicting requirements is significant.

Application of the Analysis in Horizontal Direction.
An example for conflicting requirements captured from the logistics expert is given in figure 4.

1. Relative Weighting Regarding Conflicting Relations between Requirements.

continuous conveyor		continuous conveyor
flexibility		complexity
yes		low

continuous conveyor		continuous conveyor
range of functions		maintenance effort
high		low
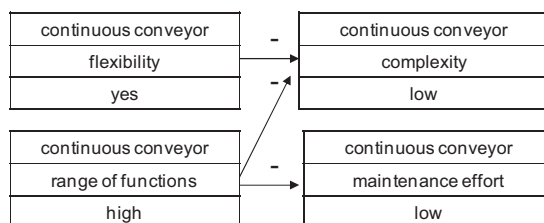
Fig. 4. Weighting of competitive requirements

Concerning this conflicting requirements a paired comparison has been performed by the logistics expert (fig. 5).

2…more important 1…equally important 0…less important	complexity/ low	maintenance effort/low
flexibility/high	1	
range of functions/high	2	1

Fig. 5. Relative requirement weighting

In order to show the effect of the developed procedure, the recorded conflicting requirements at first were weighted intuitively with absolute weightings and without a paired comparison by the logistics expert. Some weeks later the logistics expert has once again weighted those conflicting requirements with the difference that the developed proceeding was applied. The result of the comparison of the intuitive weighting which was done before via absolute weighting (ranking before) and the relative weighting (ranking afterwards) are shown in figure 6. With an intuitive weighting with absolute values the requirement for "flexibility" was at the ranking's top and after relative weighting this requirement has taken the lowest standing. The "range of functions", which is according to relative weighting the most important requirement, has only been on the second place after a first absolute weighting. With the competing requirements it was shown that "flexibility" is not more important than "complexity" (ranking before) but both are equally weighted. Furthermore, it became clear that the "maintenance costs" are not more important (ranking before) but as important as the "range of functions" So only by paired comparison the logistics expert could get an overview of the requirements whose relative weightings are the determining factor of the result of planning process.

Ranking before (absolute and intuitive weighted)

flexibility (absolute weight 4) , maintenance effort (absolute weight 4)

range of functions (absolute weight 3)

complexity (absolute weight 2)

Ranking afterwards (relative weighted)

range of functions (absolute weight 4), maintenance effort (absolute weight 4)

flexibility (absolute weight 2), complexity (absolute weight 2)
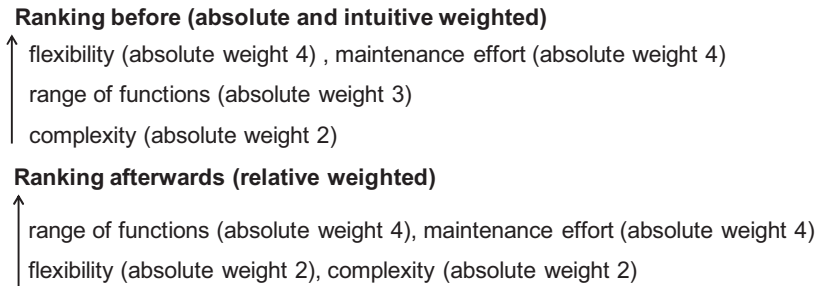
Fig. 6. Comparison of intuitive and relative weighting

2. Absolute Weighting of all Requirements Belonging to one Level.

By paired comparison in step 2, not only conflicting requirements but also partly unimportant conflicting requirements as well as requirements having no or a supporting relation, are indirectly ordered by the procedure. Although a high "flexibility" and a high "range of functions" are not in competition with each other (figure 6), the software component can derive that the "range of functions" is in a higher position than the "flexibility". The logistics expert confirmed that the new ranking achieved by the new developed procedure stronger represents his needs than the ranking resulted by intuitive allocation of absolute weightings a few weeks before. This order is basis for the customer's allocation of absolute weightings which is now following in the framework of the developed procedure. With this priority order by relative weighting the logistics expert finally has weighted the "range of functions" and the "maintenance effort" with the absolute weighting 4 and the "flexibility" and "complexity" with the absolute weighting 2 (figure 6).

4.2   Prioritization Analysis in Vertical Direction

Inconsistencies (in vertical direction) concerning the weightings of a super-ordinated requirement and its detailing requirements can be detected by means of vertical requirement relations. Therefore, after finishing the prioritization of a requirements level a consistence analysis to the super-ordinated requirement level has to be performed (figure 3). During the analysis of consistence it will be ascertained if an incompleteness of detailing requirements exists or if an under- or over-estimation of the requirements and their weightings can be assumed. Thereby, the customer can once again think over his prioritization and if necessary correct them. To check the consistence of those requirements the software component has to identify requirements which are connected to a super-ordinated requirement via vertical relations. The weightings of those detailing requirements then are compared with the weighting of the super-ordinated requirement.

Steps of the Analysis in Vertical Direction.

Within the consistence analysis the software component (performing the developed procedure) has to check the criteria shown in figure 7. While doing so in each case the

weight of one super-ordinated is compared with the weightings of its detailing requirements. First, the maximum of weightings of the detailing requirements has to be identified (**criterion 1**). If the maximum is the same as the weighting of the super-ordinated requirement, no inconsistence can be identified, because the weightings of the super-ordinated requirement and the weightings of its detailing requirements do have a sufficient similarity. In this case the prioritization analysis will be finished at this point concerning the considered super-ordinated requirement.
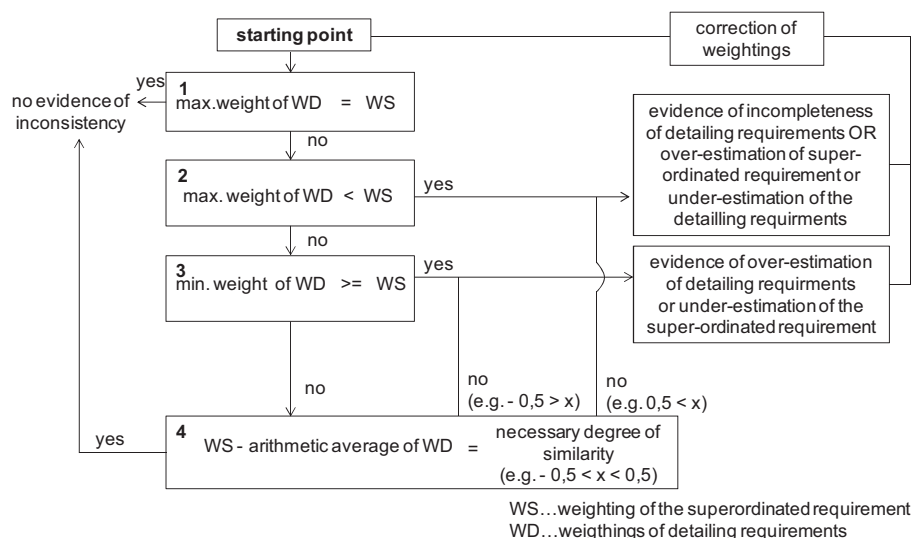
Fig. 7. Consistence analysis regarding vertical relations

If criterion 1 is not valid, the software component has to check whether the maximum weight of the detailing requirement's weighting is lower than the weighting of the super-ordinated requirement (**criterion 2**). If this is the case, this can be an indicator that further important detailing requirements are missing or that the super-ordinated requirement is weighted as too important or the detailing requirement as too unimportant. The weighting appropriately has to be corrected by the customer and the prioritization analysis will be performed again by the software component until an inconsistence cannot be proven anymore. If criterion 2 is not fulfilled, in the following the software component has to check (**criterion 3**) whether the minimum weighing is higher than/ equal to the super-ordinated weighting. In this case, it can be an evidence of an over-estimation of the detailing requirements or an under-estimation of the super-ordinated requirement.

If criterion 3 does not fit, **criterion 4** has to be proven. This happens by establishing the difference between the arithmetic mean of the weightings of the detailing requirements and the weighting of the super-ordinated requirement. This results in the extent of similarity of those weightings towards each other, which helps to identify a possible inconsistence. If the average is totally identical to the weighting of the super-ordinated requirement or if it is quite similar to it, an inconsistence cannot be con-

cluded. The necessary extend of similarity has to be determined in the requirements' gathering process before. In the given example the extent of similarity must be between -0.5 and 0.5. If the value is lower than -0.5, the detailing requirements are over-estimation or the super-ordinated one is under-estimated. The customer has to correct it and the prioritization analysis has to be repeated by the software component. This is the same, if the value is higher than 0.5 and incompleteness of detailing requirements or an over-estimation of super-ordinated requirements is indicative.

Application of the Analysis in Vertical Direction.
Figure 8 shows a cutout of the requirements model of the logistics expert. The 1st level consists of his requirement concerning a high "facilities' security" of the conveyor and the 2nd and 3th level consist of the detailing requirements of the 1st level. The results of the prioritization analysis concerning the given cutout in vertical direction are given under „weighting before". The weightings of the different levels have been compared with each other, in order to thus check their consistency. It was found that there exists a possible inconsistency between the 1st and the 2nd level.

The weighting of the "facilities' security" is 3 and the weighting of its detailing requirements is 4 in each case. The weighting 4 is both the minimum and maximum weight of the detailing requirements. The minimum weight 4 is higher than the weighting 3 of the super-ordinated requirement (high "facilities' security"), thus criterion 3 is valid. The logistics expert was informed about that. He considered his requirement for a high "facilities' security" as under-estimated. Thereupon, he corrected his weighting 3 on the value 4.

According to the consistence analysis between 2nd and 3th level, criterion 2 is valid regarding the super-ordinated requirement for a high "facilities' security concerning the transport unit". The reason for this is that the maximum weight 2 of the detailing requirements is lower than the weight 4 of the super-ordinated requirement. Due to that, there is a need for correction of weightings. According to the logistics expert a possible incompleteness of detailing requirements is not the reason for inconsistency. Instead, he sees a need for correcting of weightings of detailing requirements. In his opinion, the low weighting of "collision risk" is incorrect and should correct upwards. The "slip-resistance" and the "inclination of transport unit by shaking" are equally important for him, thus he corrected the "slip-resistance" to the weight 4. The maximum weight 4 is now equal to the weight of the super-ordinated requirement 4. Due to that the criterion 1 is effective and an inconsistency cannot be presumed anymore.
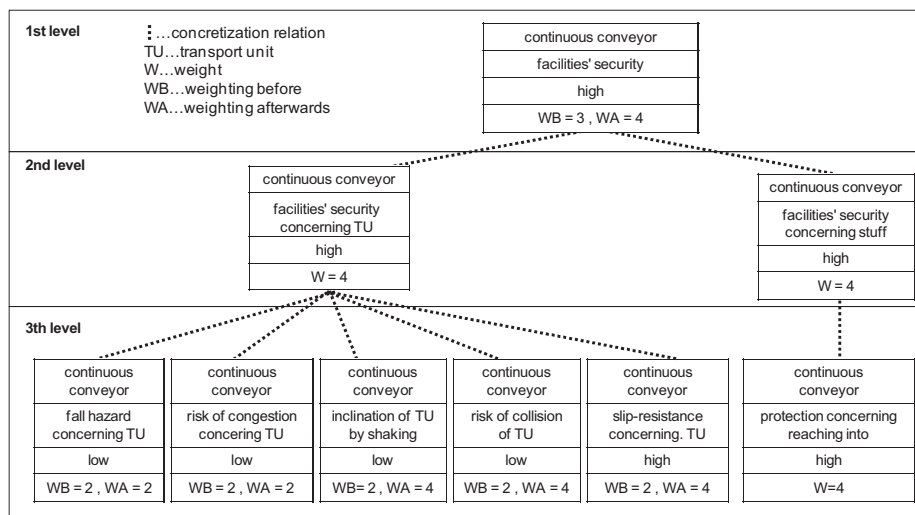
Fig. 8. Inconsistencies in vertical direction

5 Summary and Forecast

A new procedure for the prioritization of requirements has been presented, that uses requirement relations. For each new planning process it has to be decided what kind of prioritization method should be applied. If there do occur none or just a little amount of requirements relations during a planning process an absolute weighting or the method AHP should be used in order to weight requirements. In contrast, the prioritization method ANP should be used, if there exist a lot of requirements relations and the given requirements are not substituted by detailing requirements across several levels of abstraction. The new developed procedure should be applied especially in cases where a high amount of requirement relations exists and the set of requirements are substituted by detailing requirements across several levels of abstraction.

The new developed procedure enables to solve the problems of prioritization process mentioned in the introduction of this paper by using requirements relations. One above mentioned problem is the high work-intensity of the weighting process. This problem can be solved by means of the developed procedure, as it uses horizontal relations in order to determine, which part of requirements should be weighted only with absolute weightings and for this reason only with a low weighting accuracy. These are in principle, unimportant requirements as well as requirements without any relation or only with supporting relations to other requirements. The reason for this is that these kinds of requirements do have either only a low impact on the result of planning process or for these kinds of requirements a trade-off between them and other requirements must not be made. This leads to a reduction of weighting effort and due to that to a reduction of work-intensity while weighting requirements.

Moreover, the procedure enables to determine which requirement weightings have to be very precise. Horizontal requirements relations can be also used in order to de-

termine which part of requirements should be weighted with a high accuracy by means of relative weightings in advance. Conflicting requirements that are not unimportant should be compared with each other and weighted relative. The advantage of the developed procedure is that requirements, which directly compete, previously are brought into a ranking. This makes it easier for the customer to weight requirements using absolute weightings. The relative weightings of requirements will be stored and can be used during the planning process as soon as a compromise needs to be found.

Only through the concretization of requirements the customer acquires a greater awareness of what he demands from the product. The result is that he will become more aware of how important a more abstract requirement actually is in his opinion. The above mentioned risk that the weight of the super-ordinated requirement and the weightings of its detailing requirements are not consistent to each other can be checked, by means of the consistency analysis in vertical direction. Therefore requirements that are connected over vertical relations have to be identified. Due to that, the existence of a sufficient similarity of the weighting of the super-ordinated requirement and the weightings of the detailing can be proven. Thereby, an incompleteness of detailing requirements or an over- or under-estimation of requirements prioritization concerning the vertical direction can be determined.

In order to apply this developed procedure with little effort, the procedure should be implemented in the forms of a software component. To enable an automatically and thus very fast and easy identification of requirements relations, the computer-aided methods for the identification of requirements relations described in [13], [14], [15] should be part of this software component.

All in all, this procedure activates a dynamic requirements prioritization process, because the removal of inconsistencies at one point within the requirements model possibly leads to the establishment of new inconsistencies at another point. The exact steps of dynamic prioritization process and the associated risk of not reaching a consistent prioritization have to be clarified in the framework of further research activities. In doing so, it has to be examined how the correctness of the requirements weightings can be verified by means of a convergence gap as it is used during the appliance of the method Quality Function Deployment [20]. Statistical methods known from Six Sigma can be used in order to handle inconsistencies of weightings [20]. The appliance of such methods has to be proven in the future.

References

1. Maciaszek, L.-A.: Requirements Analysis and System Design, Pearson Education (2007)
2. Crostack H.-A., Mathis J.: Anforderungen an die Intralogistik – Welche Informationen sollen sie beinhalten und wie hängen sie zusammen? Berichte zum Qualitätsmanagement, Göttingen (2009)

3. Mital, A., Desai, A., Subramanian, A.: Product development: a structured approach to consumer product development, design, and manufacture. Butterworth-Heinemann (2007)
4. Ebert, C.: Systematisches Requirements Engineering: Anforderungen ermitteln, spezifizieren, analysieren und verwalten. dpunkt, Heidelberg (2010)
5. Krusche, T.: Strukturierung von Anforderungen für eine effiziente und effektive Produktentwicklung. Shaker, Aachen (2000)
6. Crostack, H.-A., Klute S., Refflinghaus R.: A holistic model for structuring requirements considering the degree of requirements' fulfilment and its implementation for data processing. In: Proceedings of the 16th International Symposium on QFD, Portland/USA (2010)
7. Humpert, A.: Methodische Anforderungsverarbeitung auf Basis eines objektorientierten Anforderungsmodells. HNI-Verlagsschriftenreihe, Paderborn (1995)
8. Heimannsfeld, K.: Modellbasierte Anforderungen in der Produkt- und Systementwicklung. Shaker, Aachen (2001)
9. Jörg, M.-A.: Ein Beitrag zur ganzheitlichen Erfassung und Integration von Produktanforderungen mit Hilfe linguistischer Methoden. Shaker, Aachen (2005)
10. Pohl, K.: Requirements Engineering: Fundamentals, Principles, and Techniques, Springer, Heidelberg (2010)
11. Rupp, C.: Requirements-Engineering und –Management – Professionelle, iterative Anforderungsanalyse für die Praxis, Carl Hanser, Munich (2002)
12. Bekkers, W., van de Weerd, I.: SPM MATURITY MATRIX: Technical Report UU-CS-2010-013, May (2010)
13. Crostack, H.-A., Kolbe, C., Refflinghaus, R.: Computer-aided method for automatic identification of effect relations between requirements on an intra-logistics facility. In: Proceedings of the 16th International Symposium on QFD, Portland/USA (2010)
14. Kolbe, C., Refflinghaus R.: Knowledge-based Concretization of Requirements in Preparation for AHP. 17th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ2011) 31st of March 2011, Essen, Germany. In: Proceedings of the Second Workshop on Requirements Prioritization for customer-oriented Software-Development RePriCo'11 (2011)
15. Kolbe C., Refflinghaus R.: Consideration of Requirements Relations during the Appliance of QFD. In: Proceedings of the 17th International QFD Symposium (ISQFD'11): Achieving Sustainability with QFD. Stuttgart, Germany (2011)
16. Refflinghaus, R.: Einsatz des Analytischen Hierarchie Prozesses zur Vorbereitung der kundenspezifischen Eingangsgrößen eines Quality Function Deployments, Sonderforschungsbereich 696: Forderungsgerechte Auslegung von intralogistischen Systemen– Logistics on Demand Technical Report 0901 (2009)
17. Gamweger, J., Jöbstl, O, Strohrmann, M: Design for Six Sigma: Kundenorientierte Produkte und Prozesse, Carl Hanser, Munich (2009)
18. Saaty, T.-L., Vargas, L.-G.: Models, methods, concepts & applications of the analytic hierarchy process, Springer, Heidelberg (2000)
19. Saaty T.-L., Gonzalez Vargas, L.: Decision making with the analytic network process: economic, political, social and technological applications with benefits, opportunities, costs and risks. Springer, Heidelberg (2006)
20. Fehlmann, T., Kranich, E.: Classification of Decisions Metrics, MetriKon 2010, http://www.e-p-o.com/downloads/classificationfordecisionmetrics.pdf, (2010)

Achieving Consensus in Requirements Engineering from the Viewpoint of Discourse Ethics

Alexander Rachmann

Hochschule Niederrhein, Centre of Excellence FAST
Reinarzstrasse 49, 47807 Krefeld, Germany
`alexander.rachmann@hs-niederrhein.de`
`http://www.hs-niederrhein.de/fast`

Abstract. Requirements engineering is commonly understood as a communicative process in which the customer and the developer achieve cooperatively consensus. This is well described by several negotiating techniques. There are several methods available to achieve consensus for ethical aspects (e.g. prioritizing requirements), but a tight integration with the findings of ethical research is still open. This paper takes some criteria from discourse ethics and applies these to consensus management in requirements engineering. The result is a rough scheme with which a developer may decide if the process of achieving consensus in requirements engineering may be "ethical".

Keywords: requirements engineering, discourse ethics, consensus, values in systems engineering, ethical discussion

1   Introduction

The concept of discourse ethics is a well established method to achieve consensus in a group. The achievement of consensus is a key activity in requirements engineering. Therefore, the combination of discourse ethics and requirements engineering seems promising. This paper makes a first step to identify criteria of discourse ethics which are relevant for requirements engineering.

Requirements engineering is described as a communicative process (chapter 2), the concept of discourse ethics is introduced (chapter 3), and an integration of both is provided (chapter 4). A small case study demonstrates the application of the before mentioned theory (chapter 5). A summary, conclusion and an outlook close this paper.

2   Requirements Engineering as a Communicative Process

It is consensus that requirements engineering is a communicative process in which the customer and the developer integrate their viewpoints by working cooperatively to achieve consensus. I.e. the customer demands requirements. Most often, the developer cannot implement the requirements precisely as formulated by the stakeholders. The developer is constrained by scarce resources such as time and budget, resulting in

conflicts [7]. Most of these conflicts arise due to different priorities assigned to the requirements by the different stakeholders. For each of those conflicts, a consensus must be achieved. In every project there will be at least a handful of conflicts of different natures: economical, social, technical, etc. conflicts. E.g. an economical consensus will be met through the usage of economical methods: A conflict concerning on what actions the disposable money should be spent, will be solved by calculating different variations of investment.

To achieve consensus, customer and developer agree upon a compromise that meets the given constraints. The compromise is rarely symmetric: most often the customer has the power to outvote the developer. However, methods for solving economical or technical conflicts, even social conflicts, are well discussed and often applied. Ethical conflicts are less well understood, despite the fact that there are some elaborate approaches, e.g. [9], [18], or [25].

The requirements engineer differentiates three core activities in his work [19]:

- Elicitation: the requirements are gathered from each (group of) stakeholder(s).
- Documentation: the elicited requirements are written down.
- Negotiation: the documented requirements are usually conflicting, resulting from the different viewpoints of the stakeholders. It is a crucial activity in requirements engineering to mediate these conflicts.

Requirements Engineering cooperates conflict solving techniques in its core activity "negotiation" [19]. Therefore the negotiating activity is the most relevant one for discourse ethics.

One point to negotiate about in almost every project is the prioritization of requirements: A (group of) stakeholder(s) may judge a requirement more important than a different (group of) stakeholder(s). Each judgment is based upon a reason, made by the individual stakeholder. In this paper, only reasons are focused that have a moral background.

3   Discourse Ethics

Ethics is the scientific discipline that focuses on moral actions. BIRNBACHER states two basic approaches in ethics: approaches that focus on which norm to follow (normative ethics) and approaches that focus on the identification of the norms to follow (method ethics) [5]. One of the best known and most influential method ethics is the discourse ethics by HABERMAS and APEL [3], [10], [11].

Discourse ethics introduces a communicative method to find consensus in a group. It focuses not just system development, but rather all human interactions. It builds upon the values of the *Aufklärung*, as stated by Kant: "Have the courage to use your own understanding" [14].

Throughout time and authors there is no common understanding on how to find a consensus, but all approaches to discourse ethics share two principles D (principle of discourse ethics) and U (universalization) [10], [12]:

- (D) Only those norms are valid that are approved by the participants of a discourse. In terms of RE, a "participant of a discourse" would be called a stakeholder.
- (U) All stakeholders accept the effects and the side effects of the approved norms.

If both principles are valid, the results of a discourse (e.g. the consensus) are morally justified. For both principles to be valid, three preconditions must be met, according to BIRNBACHER [4]:

1. Equality: Each participant recognizes the other participants as equal debater.
2. Freedom of choice: Each participant has the right to choose his "goal in life", as long as those goals in life do not conflict with the goals of the others.
3. Willingness to discuss: Each participant is willing to discuss the ethical assessment (of the requirements).

However, the principle of discourse ethics, the universalization and all three preconditions are idealistic. Their realization is difficult due to the complex hierarchies and exercises of power in society. A main problem in the realization of the theory of discourse ethics is the question, how discourse ethics can be effective in an environment with low moral aims, e.g. an environment where no discourse takes place [5], [10].

By understanding the system development process as a communicative process, all above mentioned properties of discourse ethics apply also to the development process, i.e. to requirements engineering. Weber-Wulff et al. customize a scheme for an ethical discussion to the needs of a software developer, divided in four steps [27]:

1. analysis of the situation,
2. analysis of the conflicts,
3. application of relevant norms, and
4. assessment.

The main objective of WEBER-WULFF ET AL. is the education of computer science students, not the application of the scheme in industry. Due to this objective, the scheme is not located in a development process. Requirements engineering is best suited for being such a container, mainly the process of negotiation, as proposed by POHL [19].

4   Requirements Engineering as an Ethical Discourse

As described above, there is consensus about what preconditions must be met to agree upon ethical norms. Further, the development process is acknowledged as a communicative process, and a formal scheme for discussing ethical issues is available. However, the acceptance of the development process as an ethical discourse did not yet penetrate the requirements engineering community. Fig. 1 gives an overview on how an ethical discussion is placed in the RE process: the negotiation activity of RE includes the four phases according to WEBER-WULFF ET AL. The phases depend on the realization of the universalization and the principle of discourse. These two depend upon the three preconditions as described by BIRNBACHER. From my point of view: if

the ethical discourse should be placed in requirements engineering, these three pre-conditions are the starting point.
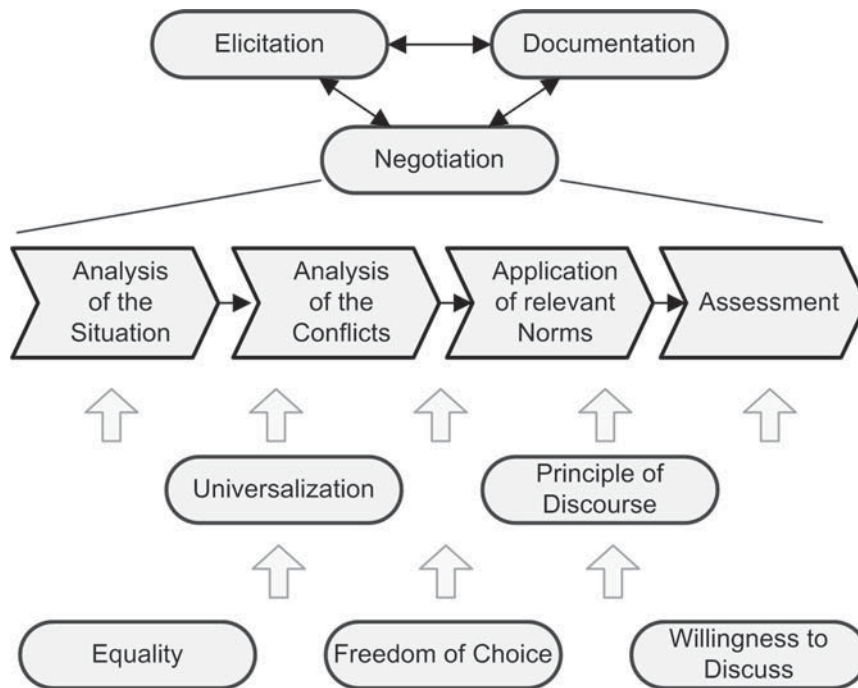
Fig. 1. Requirements Engineering as a Container for a Scheme for an Ethical Discussion

4.1 Equality

As with in society in general, there is always a hierarchy in developments projects [23]. The permission of equal rights for each project participant contradicts with the classic plan-driven engineering approaches, especially with classic customer-client relations and employer-employee relations. This may lead to the unfulfillment of the first precondition of the ethical discourse. This idealistic touch of discourse ethics is well known and often criticized [8]. However, the engineering community knows also idealized approaches very well: PARNAS AND CLEMENS describe the system development process as a goal-focused process, during which all decision are always strictly rational. In reality very few decision in development are strictly rational. Most often decisions are not even documented, such as:

- how do the goals of a system and the prioritization of the requirements relate to the ethics code of a company
- how do the goals of a system and the prioritization of the requirements relate to the ethics code of the user
- how do the goals of a system and the prioritization of the requirements relate to the ethics code of the developer

The description of an engineering process is nearly always idealized and rational. The real process is never rational. However, it still makes sense to use idealized processes, because the developer needs a guiding light to foresee undesirable trends [16].

The opposition in this discussion is headed by POPPENDIECK AND POPPENDIECK: However, they do not criticize the fact that the process is idealized, but rather in what way it is idealized. They see the agile methods as a way to better idealize the development [20].

From my point of view it is ok, that the scheme for the ethical discussion is idealized. The same arguments that hold for idealizing the development process as a whole also account for the idealized ethical discussion. However, the developers need to be aware, how the three preconditions, the principle of discourse ethics and universalization influences their work – or rather, how even the non-observance influences their work.

4.2   Freedom of Choice

HABERMAS speaks of the *Zielwahlfreiheit*, meaning the freedom of choice to choose his own goals in "life". The work of the system developer is most often not his whole life. Decisions that are made during the development process do rarely interfere with the goals in the developers' life. From this point of view, one may ease this precondition: One may regard here to a formulation such as "job-relevant goals".

The members of an organization, say a company for software engineering, most often work for other companies, i.e. as a consultant or programmer. Often, it is unclear, which goals the developer follows: his own, as an enlightened person, or the goals of the company. In a perfect situation, the goals may be the same. In the real world, the goals may differ. It is crucial to understand whose goal the developer follows. There is no simple way to determine what goals are always to follow – first, the developer has to understand, what his goals, his companies goals and the customers goals are. Obviously, if all goals are in accordance, this is the easiest variant. However, if the goals of an organization (be it the customer or someone else) are not in accordance with the developers goal, the developer has to take a position.

A very famous decision of this nature was made by WEIZENBAUM: He developed the first chat-bot named ELIZA. As a case study, WEIZENBAUM used phrases from psychotherapy in ELIZA [28]. Therefore, a human chatting with ELIZA may have had the feeling to talk to a real therapist. As this idea was absorbed by a few therapists for their day-to-day work (see [6]), WEIZENBAUM decided to stop his work on ELIZA. This was an ethical decision in contrast to his scientific community [29] [30].

4.3   Willingness to Discuss

Both freedom of choice and equality of the group members are not sufficient for a discourse to happen. Also, all participants must be willing to discuss ethical issues. For the developer and the customer are two different aspects relevant:

- The participant needs competence, expertise, authority and capacity to work in the project as well as incentives which can only be received in this exact project [15]. Besides, the participant needs the rhetorical skills to express his opinion and understand the arguments and opinions of the other participants.
- Taken as a given that the developer is motivated to work in the project *per se*: The developer has to be aware that his actions have an influence on ethical aspects. The well known attitude "I just want to develop a system, and not talk about ethics and philosophy" may hinder the discourse.

5 Case Study: Care for Elderly People Regarding Privacy and Control

A small case study, loosely based on the best practice described in [21] and [22], may clarify the application of the above mentioned theory. In this case study, a requirements engineer works in a project to develop a system for telecare of elderly people. The to-be-developed system consists of a set of sensors in the home of the customer. The sensors monitor the biosignals of the customer and recognize thereby emergencies in his life. In the case of an emergency, the care provider is contacted by the system and may help the customer.

Some real life projects concerning this objective are documented in [13], [17], [24] and [26].

5.1 Preconditions to the Ethical Discourse

At first, the requirements engineer may assemble the stakeholders: The care providers, the engineers of the technical system, and a delegate of the end-users. (In real life there may be even more stakeholders, but for the sake of the focus of the paper, one may restrict to this three stakeholders.) Then, the requirements engineer would check if this group of stakeholders fulfills the precondition for an ethical discourse: In this case study, one may take for granted, that the stakeholders are equal in their rights, they are free to choose their goals and they are willing to discuss.

As in most projects, the goals of the to-be-built system are discussed. Possible goals are:

- The care provider wants to help the end-user and thereby take a small portion of control of the life of the end-user.
- The end-user wants the help of the care provider, but at the same time does not want any inference in his privacy.

In this case, the both goals control and privacy are in a spectrum: Too much control of the care provider limits the privacy of both. At the same time, an untouched privacy takes away the control of a care provider. In the wording of requirements prioritization: The stakeholders assign different priorities to the two goals.

5.2 Prioritization of the Goals in an Ethical Discourse

As both goals are approved by both parties, there must be a prioritization of the goals (see fig. 1, "analysis of the situation" and "analysis of conflicts"). The reasons for the prioritization are based on the moral point of views of the stakeholders ("application of relevant norms"). The achievement of a consensus will bring up a system that is consistent with the care provider and the end-user (being deputized in the project) ("assessment").

The result of such a project (i.e. ethical discourse in requirements engineering) may be a system that

1. supports an elderly to keep living in his home (emphasis on privacy),
2. supports the care provider to support the elderly (emphasis on control),
3. and lets the end-user decide when and which data is being transferred to the care provider (emphasis on privacy).

To implement these three requirements, a system is needed that offers at least two different points in time to transfer data from the home of the end user to the care provider. One may think of these two points in time: The earlier data is transmitted, the more the privacy of the person is constricted and the control of the care provider is extended. The later data is transmitted, the privacy of the elderly is preserved, but at the same time, the control function of the care provider is restricted.

6 Summary

In this paper I argued that requirements engineering, as a communicative development process, is in its nature very similar to the theory of discourse ethics. Both rely on the communicative act of agreeing upon a compromise in a constrained and conflict-tainted environment, both are idealized and never strictly rational. A small case study showed the application of the discourse theory.

However, the case study disregards problems of a real-life project: It is not likely to change the hierarchy-tainted reality in a company and thereby fulfilling the preconditions for an ethical discourse. Rather, in my humble opinion, the first step should be to teach the parallels of requirements engineering with discourse ethics to the system developer, i.e. the requirements engineer. This may create thoughtfulness for his actions. This alone will make him aware of power hierarchies in projects [8]. Hopefully, requirements engineer will therefore state their ethical viewpoints and thereby start an ethical discourse. By having a formal discussion scheme in mind (see above), they can be guided through the discourse. With this background, the achieved consensus (here: concerning the prioritization of the requirements) in a project may be called morally justified.

7 Discussion at the RePriCo-Workshop

The assumption for the discussion was that there is a gap between the theory of discourse ethics and the practice of requirements engineering: Due to the fact that the three preconditions of the ethical discourse are rarely fulfilled, there can be in theory no morally justified results. However, the conception of the work of the requirements engineer is different, since a lot of results of the work of requirements engineers are in accordance with the moral views of a lot of people. From this point of view, two questions were given to the participants of the RePriCo-Workshop along with pen and paper for the metaplan technique:

- Is there an ethical discourse in "real life RE"?
- What are the factors that influence the discourse in "real life RE"?

The participants wrote their opinions and estimation on papers; the moderator of the discussion collected these papers and structured these answers. The concluding discussion explained the answers in more depth, resulting in several points:

- The work in a project is mostly influenced by a hierarchy of power and money, thereby hindering an ethical discourse. However, several participants pointed out that there is in fact a discourse in a typical requirements engineering project. Also, the influence of different cultures and application domains was pointed out.
- The importance of a market selection was discussed and by several participants supported. E.g. the requirements engineer is constructing a product without regard to a specific moral value. By giving the customer the choice to buy or not to buy a product, only those products establish that meet the moral values of the customers. This builds mainly upon the assumption that the rules of a free market are also moral values (for a similar discussion in the academic business administration discipline see [1, 2]). However, the case of market failure or imperfect market competition (e.g. monopolies) was shortly discussed. The resulting missing selection concerning moral requirements was pointed out.
- The involvement of end-user/customers is a lack of requirements engineering, even though the involvement of advocates of end-users is a viable option. If neither end-user nor advocates are present in a requirements engineering project, it turns out to be a situation where the end-user "has no rights at all".

References

1. Albach, H.: Betriebswirtschaftlehre ohne Unternehmensethik! Zeitschrift für Betriebswirtschaft 75, pp. 809-831 (2005)
2. Albach, H.: Betriebswirtschaftslehre ohne Unternehmensethik – Eine Erwiderung. Zeitschrift für Betriebswirtschaft 77, pp. 195-206 (2007)
3. Apel, K.-O.: Grenzen der Diskursethik? Versuch einer Zwischenbilanz. Zeitschrift für philosophische Forschung 40, pp. 3-31 (1986)
4. Birnbacher, D.: Habermas' ehrgeiziges Beweisziel – erreicht oder verfehlt? Deutsche Zeitschrift für Philosophie 50, pp. 121-126 (2002)

5. Birnbacher, D.: Analytische Einführung in die Ethik. Berlin, New York (2007)
6. Colby, K. M., Watt, J. B., Gilbert, J. P.: A Computer Method of Psychotherapy: Preliminary Communication. The Journal of Nervous and Mental Disease 142, pp. 148-152 (1966)
7. Fleischmann, K., Wallace, W.: Value Conflicts in Computational Modeling. Computer 43, pp. 57-63 (2010)
8. Flyvbjorg, B.: Habermas and Foucault: thinkers for civil society? British Journal of Sociology 49, pp. 210-233 (1998)
9. Friedman, B.: Value Sensitive Design. In: Encyclopedia of human-computer interaction. pp. 769-774. Great Barrington (2004)
10. Gottschalk-Mazouz, N.: Diskursethische Varianten. Deutsche Zeitschrift für Philosophie 50, pp. 87-104 (2002)
11. Habermas, J.: Theorie des kommunikativen Handelns. Band 2: Zur Kritik der funktionalistischen Vernunft. Frankfurt am Main (1981)
12. Habermas, J.: Diskursethik – Notizen zu einem Begründungsprogramm. In: Moralbewußtsein und kommunikatives Handeln. pp. 53-125. Frankfurt am Main (1983)
13. Khaled, B., Rumm, P. and Lukowicz, P.: AiperCare – ein interaktives Monitoring-System für Personen mit neurologischen Einschränkungen und deren Umfeld. In: Proceedings of the 5. Deutscher AAL-Kongress Technik für ein selbstbestimmtes Leben. 24.-25. January 2012 in Berlin. VDE Verlag (2012)
14. Kant, I.: An Answer to the Question: What is Enlightenment? http://www.english.upenn.edu/~mgamer/Etexts/kant.html (1784)
15. Lüschow, F.: Arbeit in der Linienorganisation – Arbeit im Projekt: zwei Welten begegnen sich. In: Vortrag auf dem 24. internationalen Deutschen Projekmanagement-Forum 2007 (GPM) in München. (2007)
16. Parnas, D. L., Clements, P. C.: A rational design process: How and why to fake it. IEEE Trans. Softw. Eng. 12, pp. 251-257 (1986)
17. Pflüger, M., Kroll, J. and Steiner, B.: Automatische Notfallerkennung durch optische und akustische Sensoren. In: Proceedings of the 5. Deutscher AAL-Kongress Technik für ein selbstbestimmtes Leben. 24.-25. January 2012 in Berlin. VDE Verlag (2012)
18. Pietsch, W.: Ethical Product Management Employing QFD. In: Proceedings of the 15th International Symposium on Quality Function Deployment, Monterrey, Mexico, October 22.-23.2009. pp. 45-52. (2009)
19. Pohl, K.: Requirements Engineering - Fundamentals, Principles, and Techniques. Berlin (2010)
20. Poppendieck, M. B., Poppendieck, T. D.: A Rational Design Process – It's Time to Stop Faking it. http://www.leanessays.com/2010/11/rational-design-process-its-time-to.html (2000)
21. Rachmann, A.: Telemonitoring-Dienstleistungen in der Altenhilfe unter besonderer Berücksichtigung von Kontrolle und Privatsphäre. Working paper 11-81 of the chair Wirtschaftsinformatik at the Ruhr-Universität Bochum (2011).
22. Rachmann, A.. Referenzmodelle für Telemonitoring-Dienstleistungen in der Altenhilfe. In: Proceedings of the Modellierung 2012, 14.-16. March 2012 at the Otto-Friedrich-Universität Bamberg. GI-Edition Lecture Notes in Informatics P-201, pp. 219-234 . Gesellschaft für Informatik, Bonn (2012).
23. Rittgen, P.: Negotiating Models. In: Advanced Information Systems Engineering. LNCS vol. 4495. pp. 561-573. Springer, Heidelberg (2007)
24. Rodner, T., Floeck, M. and Litz, L.: Inaktivitätsüberwachung und Alarmhandling zur Verringerung von Fehlalarmen. In: Proceedings of the 4. Deutscher AAL-Kongress Innovative

Assistenzsysteme im Dienste des Menschen – Von der Forschung für den Markt. VDE Verlag (2011)

25. Ropohl, G.: Ethik und Technikbewertung. Frankfurt am Main (1996)

26. Spehr, J., Gietzelt, M., Wegel, S., Költzsch, Y., Winkelbach, S., Marschollek, M., Gövercin, M., Wahl, F., Haux, R. and Steinhagen-Thiessen, E.: Vermessung von Gangparametern zur Sturzprädikation durch Vision-und Beschleunigungssensorik. In: Proceedings of the 4. Deutscher AAL-Kongress Innovative Assistenzsysteme im Dienste des Menschen – Von der Forschung für den Markt. VDE Verlag (2011)

27. Weber-Wulff, D., Class, C., Coy, W., Kurz, C., Zellhöfer, D.: Gewissensbisse: Ethische Probleme der Informatik. Bielefeld (2009)

28. Weizenbaum, J.: ELIZA – A Computer Program For the Study of Natural Language Communication Between Man And Machine. Communications of the ACM 9, pp. 36-45 (1966)

29. Weizenbaum, J.: Alptraum Computer. Die Zeit 43 (1972)

30. Weizenbaum, J.: Computer Power and Human Reason. London (1984)

Requirements Negotiation
in Consideration of Dynamics and Interactivity

Andreas Reiser[1], Benedikt Krams[2], Mareike Schoop[1]

[1] Information Systems I, University of Hohenheim, 70593 Stuttgart, Germany
`andreas.reiser@wi1.uni-hohenheim.de, m.schoop@uni-hohenheim.de`
[2] Department for Business Administration and Information Systems II, esp. Business Software, Universität Stuttgart, 70147 Stuttgart, Germany
`krams@wi.uni-stutgart.de`

Abstract. Due to conflicts which arise between customers and developers in software development it is worthwhile to regard requirements engineering (RE) from a decision making and communication perspective. If comparing RE with negotiation theory there exist important similarities between both perspectives. This paper introduces a negotiation-centered view on requirements elicitation and focuses on two main aspects, namely process definition and the role of asymmetric information. A software platform offering different methods of requirements elicitation, communication support including semantic enrichment, and negotiation support in general will be introduced to highlight a new approach to requirements negotiation. After introducing several steps for improvement of the software platform a conclusion in the context of the process definition in RE and asymmetric information is given.

Keywords: Requirements engineering, requirements negotiation, requirements elicitation, requirements prioritization, preference elicitation, dynamic preference elicitation, decision support, communication support, Negoisst

1   Introduction

Software development starts with the phase of requirements elicitation and analysis. In an idealised view, customers know their complete requirements, are able to utter them in a way that is directly understandable for the software development team who will develop the product according to the requirements, and no conflict between customer and developer will occur during the process.

However, the real world is not an ideal one. Firstly, from a *decision making perspective*, customers do not always know their complete requirements at the very beginning of the joint elicitation process. Instead, the participants will exchange information during the whole requirements engineering (RE) process and constantly refine or adapt their preferences to their current knowledge. Second, from a *communication perspective*, the participants are sometimes not able to utter what they need in a clear, direct, and structured manner. Additionally, conflicts that might occur during the software development process between customer and development team need to be addressed and, if possible, solved.

The two above-mentioned problem types are typical issues stemming from the dynamical and interactive character of requirements elicitation and analysis in contrast to one-shot elicitations of other domains like market research or business intelligence. Moreover, these two aspects show some important similarities of the requirements engineering domain compared to current research activities in negotiation theory. Newer developments in negotiation theory tend to propagate a dynamic perspective on information exchange and preference elicitation [10], [14]. We therefore introduce a negotiation-centered view on requirements elicitation and highlight the requirements for communication- and decision support in this iterative process.

Our research follows a design science approach [5]. We first focus on assessment of criteria for a communication-centered requirements engineering process. Second, we will introduce a software artifact as a first evaluation for an iterative refinement. To this end, the paper introduces a software platform offering different methods of requirements elicitation (to help customers find out and express their requirements and preferences), communication support including semantic enrichment (to discuss needs in a flexible yet structured way), and negotiation support in general (to deal with different goals and enable conflict management). This article mainly describes work in progress; therefore we will particularly highlight some of the main challenges, namely the process design and the influence of asymmetrical information.

2   A Negotiation-centered View on Requirements Engineering

Requirements engineering in general consists of the requirements analysis and requirements management, is a "(…) cooperative, iterative, and incremental process" [13], and can be seen as multi(bi-)lateral meta-negotiations, i.e. negotiations about the composition of specific negotiation contents (requirements). In other words, we do not conduct negotiations about the exchange of goods directly, but about the specifications of goods. This negotiation-centered perspective on RE highlights certain aspects of the RE-process that are not covered by other models yet. In this paper, we want to address two aspects that can be beneficial in many RE situations. First, we discuss the differences in the *process* itself, especially the communication-centered perspective in negotiations and the interaction with decision support. Second, we address the problem of *asymmetric information* and some suggestions how to deal with this potential problem. Referring to [2], we will use the term requirements negotiation (RN) for this view on RE.

2.1   The Process of Requirements Negotiation

RE usually involves a lot of resources (personnel, time, money, etc.) giving rise to the need of a good administration for the whole process. One of the main challenges is to effectively catch and align the requirements of all stakeholders involved. This task most likely becomes a difficult process since this involves considering several procedural and behavioral attributes at the same time. In order to reduce this cognitive complexity, several requirements elicitation and/or analysis procedures have been

proposed for the domain of requirements engineering, either as task-specific models or general frameworks (for an overview see [4]). Most of these procedures are mainly developed as process models for various requirements engineering activities. This basically enables a holistic approach to RE where the process will be shaped by the applied model. The benefit of using these model-centered approaches is a uniform, well-ordered and clear line of action for all stakeholders. On the other hand, this requires the participants to learn a specific RE-model beforehand and adapt their way of thinking to meet this process.

This raises the question whether it is always the best option to determine the RE-process by providing a full-blown and fixed process definition. A negotiation-centered approach can solve this problem by supporting the communication and decision making instead of providing a strict path of actions. However, this doesn't mean just to provide a communication and/or decision support platform. Instead, several RE-inherent tasks must be considered for providing the right support tools for the task at hand. We identify several needs of support techniques that are common to negotiation in general and for the domain of requirements negotiations in particular.

Parallel to negotiations, the communication and decision making process in RE can be divided into three phases: preparation, negotiation and settlement [9]. At the beginning of the RE-process, the participant has to prepare himself for the later exchange of information which – from a stakeholder's perspective – mainly involves the formulation of goals, needs, arguments and the demand for information. In this phase, we need a rather *individual support* for the *requirements elicitation*. In the second phase, the stakeholders will exchange information and try to align their preferences which require *common support* for *communication* and *decision making* in order to build optimized, joint solutions. In the last phase, the generation of a settlement, the participants also need communication and decision making support, as well as additional *document support* e.g. for building legal requirements specification.

The use of different support tools for different phases of the RE-process also separates the RE-process models from the RN-approach: instead of actively forcing a particular procedure, the RN-approach focuses on the communication of the stakeholders with additional, optional support tools. Therefore, the main concern of RN is not to choose the right process model, but to provide the right tools at the right time.

2.2 Decision Support in Requirements Negotiation and the Role of Asymmetric Information

Following the above-mentioned negotiation phases, decision support is present in all three phases of the RN-process, for individual elicitation as well as joint decision making. Therefore, from the various classes of decision support systems, a tool for RE needs to enable personal support as well as group support [6]. In addition, drawing a clear distinction between requirements elicitation and analysis enables the participants to choose from a variety of elicitation methods which best fits the needs for the specific RE task.

In the preparation phase, we will individually elicit the preferences of every stakeholder involved which can be done either using qualitative or quantitative assess-

ments. However, from a decision support point of view, a usual way is to provide a method to quantify all these requirements as a unified measurement (e.g. scoring systems, utility models, etc.) allowing for detailed analysis of the stakeholder's preferences (e.g. comparison, aggregation and trade-offs).

In the negotiation and settlement phase, the preference model can then be used for a wide range of analytical support tools, e.g. graphical representations, verbal suggestions or numerical indicators. This analytical support can be further divided into asymmetrical support tools for measuring the individual performance and symmetrical support tools for a joint optimization [14].

Now, if we treat RN as communication-centered process, how can we link the elicitation of individual requirements to this process? There are two important aspects of communication in negotiation that need to be considered in the progress of elicitation [20]. First, RN is an *interactive* process where people exchange ideas, thoughts and arguments, in other words, we have to consider formal and informal communication aspects [17]. This communication process involves the exchange of information which in turn influences the preferences of the participants. Second, RN is a *dynamic* process, not a one-shot proposal. Therefore, the preferences must be adapted to the current information level.

As these two aspects imply, considering asymmetrical information is a critical requirement for using a communication-centered RE process. More precise, asymmetrical information has a high impact on the whole RN-procedure:

- In contrast to normative economics, the participants in a RE-process do not start with enough information to make a decision. Therefore, this must be considered in the initial requirements elicitation. The initial preference model may contain errors stemming from individual assumptions, beliefs and thoughts. As a consequence, the elicitation procedure must support an error rate or fuzzyfication.
- As with communication exchange advances so comes new or updated information. It is very likely that this information also influences the assessment of the requirements. Therefore, the elicitation method must be capable of catching these changes in the evaluation of the requirements. This can be done in two ways, either automatically or interactive. For an automated assessment of requirements, the decision support system must provide information about divergences in the preference model, e.g. by measuring consistency. For an interactive assessment, the user provides additional information for discovering divergences. This can be done e.g. by asking the user about his current level of information for single requirements.
- If a divergence of the initial and current preference model is discovered, the evaluation of requirements must be updated to the current information level. This can also be done automatically or interactive. An automated approach will need to process the new information and quantify them for the individual evaluation of the participant. On the other hand, the requirements can also be re-elicited by the user himself. However, it does not make sense to re-elicit every single issue since this will bind additional resources. Instead, an appropriate method must be capable of only catching necessary changes in the preference model.

In the following chapters, we will introduce a software prototype which considers both, process and information requirements for an RN-model.

3 A Scenario for Requirements Elicitation and Requirements Prioritization

As conflicts are the trigger for negotiations (cf. chapter 1) we envision a short scenario to discuss ex-ante and dynamic preference elicitation as well as analytical support in the context of requirements negotiation.

Initial point is the wish to develop a piece of software (e.g. an add-on to a text program to compile bulk letters) which needs to be done due to reasons out of focus in a distributed software development team. This team consists of a group of customers and a group of developers which are located in different continents. Responsible project managers of both groups decided to use a collaboration tool because of spatial and temporal allocation of the groups.

The software platform enabling and supporting multi-professional interactions by means of decision support and communication support in such a scenario is called Negoisst [18]. This platform integrates three different types of support, namely decision support, communication support and document management. Although all three types are relevant in RE process, in this paper, we will focus on the communication and decision support components.[1]

Requirements negotiations are iterative and dynamic processes. Dynamic, because arguments, beliefs, and thoughts are exchanged between all participants which will then influence the process itself. In this process, the negotiators try to reach a joint outcome that best fits the needs of all participants, i.e. to reach an integrative solution (win-win situation, cf. e.g. [2]). Iterative, because RE is a communication-driven process where the participants iteratively exchange arguments in order to converge to a joint solution.

Therefore, decision support in requirements engineering is not a one-shot assistance, but needs to cover the whole process. In Negoisst, decision support consists of two steps (for a more detailed description see [14]): The first step is a preference elicitation process to compute an individual utility model, using compositional and/or decompositional preference elicitation methods adapted from Multi-Attribute Utility Theory (MAUT). MAUT serves as the base model in many preference elicitation domains e.g. market research or business intelligence. The benefit is that these models follow a common preference concept that can be easily understood and interpreted by all participants.

In the second step, the stakeholders are presented with a wide range of analytical support tools based on the user's individual utility model.

---

[1] For a detailed description of the support types communication support and document management see [17].

3.1   Ex-ante Preference Elicitation

The aim of preference elicitation is to get a standardized representation of someone's real preferences (which are not directly ascertainable in general). In this prototype, we will represent the user's preferences by generating a utility model. The reason for this approach is the uncomplicated traceability and interpretation which only requires little training of the stakeholders. Additionally, a utility-based approach enables an easy solution for comparing and aligning the preferences of the participants. In a requirements engineering process, we therefore need to evaluate the individual importance for every single requirement (e.g. database design: accessibility vs. high transaction isolation), as well as the preferences for the individual options of a single requirement (e.g. isolation levels {Repeatable Read; Read Stability; Cursor Stability; Uncommitted Read}).

The question arises, which method to use for the elicitation of preferences. In general, this choice should be based on the cognitive complexity of the decision at hand. In a well-structured, fully specified decision problem, the participant can therefore explicate his/her preferences directly. On the other hand, if the decision-maker will be confronted with an un-structured and/or only partially specified decision problem, he/she cannot conduct rational decisions. Instead, those decisions will be based on non-cognitive behavior, basically built on fuzzy knowledge and various unconscious factors [12]. In this case, someone would use indirect preference elicitation methods, such as Conjoint- or Discrete Choice-Models. In the domain of requirements engineering, we will usually find both types of behavior. We therefore suggest a hybrid preference elicitation process which covers various types of behavior.

The current Negoisst system is based on a direct self-explicated approach based on a hybrid conjoint model [11], [1] and an Analytical Hierarchy Process (AHP, in [15], [16]) as an alternative method. The choice, which preference elicitation procedure to use mainly depends on the initial RE situation. If there exist some predefined alternatives (e.g. which software to buy), AHP will be the preferred option. On the other hand, if the RE-process mainly focuses on discovering and aligning requirements, one should prefer a hybrid conjoint model instead. The preference model considers two different part worth types, i.e. numeric attributes (with which continuous codomains can be represented, being limited by a best case and a worst case value, e.g. a price) and categorical attributes (with which discrete codomains are represented, e.g. a finite set of colours such as black, blue, orange). The utility function is computed accordingly and the utility values are displayed for each message written and received. In this prototype, we use a linear-additive utility model.

Figure 1 shows the self-explicated preference elicitation method. This method is also used for some of the hybrid elicitation methods.

Fig. 1. Self-explicated preference elicitation

Figure 2 shows an indirect preference elicitation method, namely a choice-oriented, hybrid individualized conjoint analysis. This approach holds a sufficient approximation of the user's preferences, as long as the attributes are loosely structured without a strong hierarchy.

Fig. 2. Choice-oriented, hybrid individualized conjoint analysis

3.2 Dynamic Preference Elicitation

An ex-ante preference elicitation performs well if all relevant information is known to the participants and if there is only negligible uncertainty concerning the evaluation of the requirements. On the other hand, if the preferences are likely to change during a requirements engineering process, the conventional ex-ante elicitation would lead to a considerable effort for re-evaluation of the user's preferences (for details on the costs of re-elicitation see [10]. More generally, the adaption of traditional MAUT models (mostly used in market analysis) does not work well for the dynamic, iterative character of requirements engineering.

The underlying reasons for the need of dynamic preference elicitation support can be found within two commonly cited problems of failing software development projects: "incomplete requirements" or "changing requirements" [20], [13].

The incompleteness of requirements has diverse causes such as methodological problems (lack of knowledge of methods or tools), communication problems (incomprehension, different terminologies, different backgrounds), and psychological causes (lack of trust, lack of mutual acceptance, tactics) [3]. The underlying assumption is that customers know their needs and utter them during requirements elicitation. How-

ever, many requirements are unknown or are not uttered as they seem self-evident to the customers. For example, a customer would not explicitly demand for an add-on to text program to have the basic functionality "it must be possible to write text" as this functionality of the software is expected. Such requirements are difficult to ascertain by the developer. Furthermore, requirements exist which are not expressed during requirements elicitation by customers due to a lack of knowledge of possible technologies (cf. the impact of requirements on the customer satisfaction as a triage of 'delightes', 'satisfiers', and 'dissatisfiers' by Kano [8], [13].

Two substantial causes for requirements changes need to be separated. Firstly, requirements can change as a reaction to a change in the environment (such as changing markets, business processes, or new laws). Traditional RE methods mostly try to avoid requirement changes as much as possible and if they occur, try to manage these changes by thorough formal processes. Later requests for requirements changes are interpreted as disturbances in the project which lead to corresponding time lags and/or an increase in costs. Newer RE methods (e.g. in the context of agile software development) meet these changes by employing evolutionary methods of development. Secondly, requirements can change due to technological changes. Many requirements specifications do not only contain customers' requirements but already technical product functions as solutions to implement these requirements. As information technology undergoes steady improvements changes are inevitable.

In case of incomplete information, Negoisst offers a dynamic preference elicitation method. This dynamic approach uses two complementary elicitation processes: a fast preference elicitation method for the initial elicitation and an iterative elicitation method for refining preferences during the negotiation phase. In the pre-negotiation phase, Negoisst uses a decompositional, polynomial elicitation method for fast initial approximation of a user's utility model. The second elicitation is based on the iterative offer communication process during an ongoing negotiation (or requirements engineering). Since all of the user's proposals are feasible, and, to some degree part of the aspired set of alternatives, an additional decompositional comparison of the user's offers helps to detect changes and inconsistencies in the initial utility model.

3.3 Analytical Support

The electronic support tools for analyzing the negotiator's preferences can broadly be categorized as numerical indicators, graphical representations and verbal suggestions.

(1) Numerical indicators provide a quick rating of the current offer, e.g. total utility, option valuation, utility tables etc. Negoisst thereby supports partial offer specification, that is, if someone does not specify all agenda items (requirements), a utility range will be displayed for a rough orientation. Figure 3 shows the message exchange with the total utility rating for each offer on the right.

Fig. 3. Message exchange with utility range attached

(2) Apart from this highly aggregated information, graphical representations provide a more sophisticated overview of the whole negotiation process. One of the most important distinctions of the various graphical representations is the degree of confidence concerning private utility information. A history graph (Figure 4) shows the convergence of the offer communication process using only the negotiator's own preferences. On the other hand, a negotiation dance graph (Figure 5) discloses private utility information from both negotiation sides. With this additional information, a user can easily identify inefficient outcomes in a requirements engineering process. The question as to how much private information should be disclosed depends on the type of negotiation. Since requirements engineering tends to be a highly integrative negotiation situation, this private information should be disclosed in order to gain a higher joint outcome (in terms of requirements).
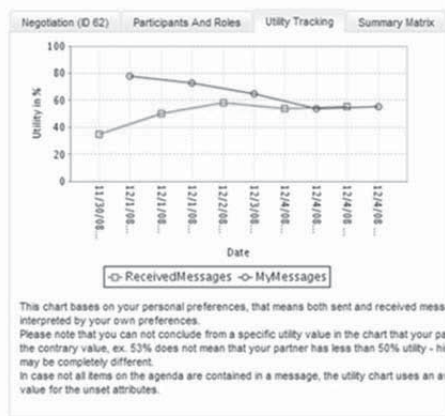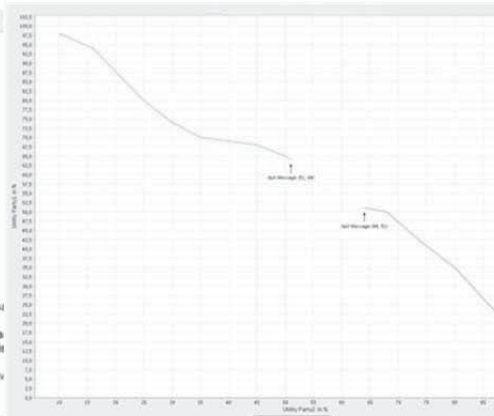

Fig. 4. History Graph   **Fig. 5.** Dance Graph

(3) Additionally, feature-rich negotiation support tools like Negoisst also uses various tools for verbal suggestions (e.g. mediation, association rule learning results, ...). Moreover, ex-post analysis tends to optimize the joint outcome, e.g. automatic search for pareto-efficient solutions.

4   Outlook

Since Negoisst was originally developed as a mere negotiation support system it is planned to extend the functionalities for requirements engineering threefold.

(1) In the case of drawing up a compulsory contract from a functional specification document conflicts might arise again due to cost and time constraints for a development project: as the implementation of a certain product function to fulfill certain customers' needs can be very complex, developers might demand some extra time or might charge higher rates. Negotiations about the final contract settings become inevitable.

(2) The customers' needs and product/quality functions are part of a document containing the functional specification which is usually set up after the elicitation and prioritization of requirements. These functions will become part of a contract between customers and developers, respectively the principal and contractor. Additionally, aspects of a prototype software requirements specification such as the scope of the software, important definitions, or constraints [7] should be considered in the RN-process. These aspects can be seen as part of an agenda, giving rise to an integration of contract management for the RN-process. With the aid of Negotiation Support Systems, blueprints of formal contracts can be prebuilt semi-automatically using semantical enrichment of the communication process [19]. This will reduce administrative overhead for the stakeholders and legal representatives.

(3) If viewing requirements engineering as a negotiation process, new or changed requirements will lead to changes of the negotiation agenda. To clarify which requirement is part of the agenda, i.e. which requirement is necessary, useful, or technically feasible, the stakeholders have to negotiate about this agenda (typically denoted as agenda-negotiation or meta-negotiation). Thus, not only contract negotiations (i.e. business negotiations dealing with requirements and their values such as price as described above) but also agenda negotiations in the meaning of negotiations concerning the addition of requirements and thus extending the agenda must be considered in the RN-process. This will also affect the deployment of decision support components since changes in the negotiation agenda will also entail changes in the preference model. Therefore, in addition to consideration of information asymmetries, an RN-system also needs to cover CRUD-operations of the negotiation agenda.

Agenda negotiations can take place upfront to set up the negotiation context but will also be a permanent part of the whole requirements negotiation process. In fact, a negotiation process in RE will be a continuous process with breaks, i.e. it will start for the initial requirements; then once new requirements have formed, a new round obviously based on the previous one will be initiated; the system will be adapted, new requirements lead to a next negotiation round based on all previous ones and so on.

5   Conclusion

This paper shows a communication-driven approach for requirements engineering in contrast to holistic process models. Recent developments in negotiation theory can help achieving not only an effective RE result (a higher joint outcome) but also an efficient and integrative RE-process. We therefore promote the research approach of requirements negotiation. In this first iteration of a software artifact, we focused on two main aspects, namely process definition and the role of asymmetric information.

Regarding the first aspect (process definition), we can see that a communication-driven approach will lead to an implicit process model instead of providing an explicit process definition. This gives stakeholders more freedom for their very own way of RE for the task at hand. Additionally, the participants do not have to learn a specific process model. On the other hand, in addition to a sophisticated communication support, one of the main challenges of RN is to provide accurate support tools at the right time. In the preparation phase, the stakeholders need comprehensive decision support tools for eliciting their individual requirements. Later, during the ongoing negotiation, the participants need additional analytical support for reaching an integrative, joint agreement. Several support tools have been presented either for individual or symmetrical analysis of the stakeholders' requirements.

One issue which should never be neglected in a RE-process is the general problem of asymmetric information. This requires special attention if requirements elicitation is not defined by the RE process. We therefore show important factors that need to be considered in a RN procedure. First, the elicitation procedure needs to support a general error for the assessment of specific requirements. Later, during the ongoing communication process, changes to the initial preference model must be discovered and then adapted to the current information level. We therefore encourage recent developments focusing on a dynamic view on decision support.

In addition to these two aspects for negotiating requirements as a task of aligning the stakeholder's individual positions, we also discussed the integration of contract negotiations and agenda negotiations. In current systems the agenda is set up before the start of the electronic negotiation. Agenda negotiations need to be developed as they provide the means to discuss the requirements themselves. Such processes need to be integrated with contract negotiations in which requirements with their values are exchanged in (counter-) offers.

References

1. Green, P.E. and Srinivasan, V.: Conjoint Analysis in Marketing: New Developments with Implications for Research and Practice. The Journal of Marketing 54, 4, 3-19 (1990)
2. Grünbacher, P. and Seyff, N.: Requirements Negotiation. In: Aurum, A., Wohlin, C. (eds.) Engineering and Managing Software Requirements, pp. 143–161 Springer, Heidelberg (2005)
3. Herzwurm, G., Schockert, S., Mellis, W.: Joint requirements engineering. QFD for rapid customer-focused software and internet-development. Vieweg, Gabler, Braunschweig (2000)

4. Hickey A.M., Davis, A.M.: Requirements elicitation and elicitation technique selection: model for two knowledge-intensive software development processes. In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences, pp.10 (2003)
5. Hevner, A.R., March, S.T. Park, J., Ram, S.: Design science in information systems research. MIS Quarterly, 28, 75-105 (2004)
6. Holsapple, C.W., Singh, M.: Electronic Commerce: From a Definitional Taxonomy Toward a Knowledge-Management View. Journal of Organizational Computing and Electronic Commerce 10, 3, 149-170 (2003)
7. IEEE Std 830-1998. Software Requirements Specification (1998)
8. Kano, N., Nobuhiku, S., Fumio, T., Tsuji, S.: Attractive quality and must-be-quality. Journal of Japanese Society for Quality Control 14, 2, 39–48 (1984)
9. Kilgour, D., Eden, C. (eds.): Handbook of Group Decision and Negotiation. Springer, Dordrecht (2010)
10. Köhne, F., Schoop, M., Staskiewicz, D.: Decision Support in Electronic Negotiation Systems - New challenges. In: Proceedings of the IFIP DSS2004 Conference (2004)
11. Luce, R. and Tukey, J.: Simultaneous Conjoint Measurement. Journal of Mathematical Psychology 1, 1 (1964)
12. McCullough, D.: Trade-off analysis: a survey of commercially available techniques. Quirk's Marketing Research Review (1998)
13. Pohl, K.: Requirements engineering. Fundamentals, principles, and techniques. Springer, Berlin (2010)
14. Reiser, A.: The use of dynamic preference elicitation for negotiations with incomplete or missing information (2010)
15. Saaty, T.: The Analytic Hierarchy Process, Planning, Piority Setting, Resource Allocation. McGraw-Hill, New York (1980)
16. Saaty, T.: The Analytic Hierarchy and Analytic Network Processes for the Measurement of Intangible Criteria and for Decision-Making. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) Multiple Criteria Decision Analysis: State of the Art Surveys, pp. 345–408. Springer, London (2005)
17. Schoop, M.: Support of Complex Electronic Negotiation. In: Kilgour, D., Eden, C. (eds.) Handbook of Group Decision and Negotiation, pp. 409–423. Springer, Dordrecht (2010)
18. Schoop, M., Jertila, A., List, T.: Negoisst: a negotiation support system for electronic business-to-business negotiations in e-commerce. Data Knowl. Eng. 47, 3, 371–401 (2003)
19. Staskiewicz, D.: Document-centred electronic negotiations. Munich (2009)
20. The Standish Group: Chatering the Seas of Information Technology: Chaos (1994)
21. Tutzauer, F.: Toughness in Integrative Bargaining. The Journal of Communication 43, 1, 46–62 (1993)

Tackling Prioritization in
Business-Process-Driven Software Development

Norman Riegel[1], Joerg Doerr[1], Oliver Hummel[2]

[1]Fraunhofer IESE, Fraunhofer Platz 1, 67663 Kaiserslautern, Germany
{norman.riegel, joerg.doerr}@iese.fraunhofer.de
[2]Software Engineering Group, University of Mannheim, 68131 Mannheim, Germany
hummel@informatik.uni-mannheim.de

Abstract. Information system (IS) development projects in enterprises are often aligned with business process improvement programs aimed at optimizing business performance. As experienced in our projects, decision makers need better support in order to enable them to assess the best way for spending the available resources. In the area of requirements engineering (RE) and release planning, prioritization is an established strategy for achieving this goal. Prioritization approaches currently found in the literature, however, do not consider all idiosyncrasies of business-process-driven software development. Hence, in this paper we analyze the suitability of state-of-the-art prioritization approaches based on the characteristics of typical business-process-driven software development projects. We identify the shortcomings of current approaches and outline our vision of how to overcome them in the future.

Keywords: Requirements Engineering, Requirements Prioritization, Requirements Elicitation, Business Process Management, Information Systems

1   Introduction

The purpose of many information system (IS) development projects is to build software to better support an enterprise's business processes and optimize business performance. To achieve this, business processes have to be identified, analyzed, and investigated regarding potential improvements. Corresponding software requirements must then be derived for the development of the supporting IS. In such a scenario, business process management (BPM) and requirements engineering (RE) are strongly intertwined. Hence, in such a setting, RE activities cannot be carried out independently, as they are affected by business process reengineering and improvement efforts [2]. We use the term business-process-driven requirements engineering (BPRE) to describe an RE process that is based on the business processes and workflows of an enterprise and derives software requirements from them in a systematical manner (cf., e.g., [2][3]). During several of our industry projects, we recognized that decision makers had difficulties in applying common prioritization approaches in BPRE settings because they could not be tailored for this context. This

led to wasted time and effort spent on numerous (RE) activities of minor importance: Conducting a workshop on optimizing a business process that later turns out to contribute only low-priority software requirements is an example of such an unpleasant situation. The remainder of this paper is structured as follows: Section 2 sketches the characteristics and challenges of BPRE projects and maps them to requirements for an appropriate prioritization approach; in section 3, existing prioritization approaches are assessed against these requirements; section 4 discusses the outcome and presents our vision for a solution idea, and section 5 summarizes our contribution.

2   Characteristics and challenges of BPRE

Our previous work on prioritizing requirements in software development (cf. [2]) already presented some ideas on how to tackle the challenge of requirements prioritization in the context of BPRE. Following up on that, in this section we sketch the typical characteristics and challenges we have identified for requirements prioritization in this context. Based on this, we systematically derive requirements that a prioritization approach for BPRE should support in order to be applicable in such a setting (in accordance with the general selection process described in [7]).

Issue (1): Typically, BPRE projects are characterized by *high complexity* – even in small and medium-sized enterprises it is not uncommon to have different business areas containing several dozen business processes, which in turn consist of numerous business activities that need to be considered for optimization [2] (Fig. 1 exemplarily shows a hierarchy of different issues of interest). It is obvious that it does not make sense to refine each feasible software design, as the required effort would simply be too large. Eliciting the requirements in an efficient way (i.e., in the optimal order) while also handling the challenge of the *large number of possible system designs* [6] (e.g., transformation from as-is to to-be or different levels of automation) is a challenge for every requirements engineer.

> → **Req. 1:** The prioritization approach should *guide the elicitation* in order to make it as efficient as possible, while not lowering final system quality at the same time.

Issue (2): During requirements elicitation, business processes and included activities are analyzed systematically and more detailed requirements are derived [6]. Thus, the requirements *form a hierarchy*, i.e., dependencies are created from abstract business processes down to detailed system functions and other additional requirements [2].

> → **Req. 2:** The prioritization approach should support *continuous / consistent prioritization* across all levels of abstraction *by supporting hierarchies* and ideally also other dependencies.
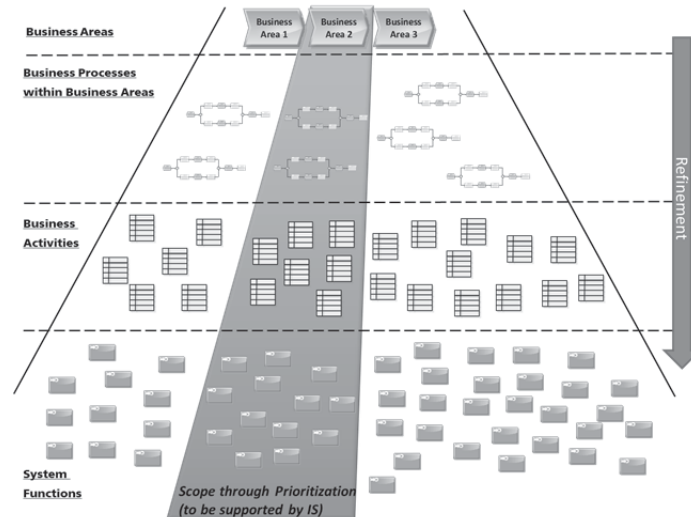
Fig. 1. Guidance for requirements elicitation using prioritization.

Issue (3): As a hierarchy of requirements is created, *different requirements types* (i.e., requirements concerning different issues) on different levels of abstraction are elicited (e.g., business processes, business activities, derived system functions, etc. [2]).

> → ***Req. 3:*** The prioritization approach should take into account *the idiosyncrasies of the requirements types* (e.g., information content that is available on a particular level of abstraction).

Issue (4): The interests of *several stakeholders and roles* within the business processes have to be considered. This does not only include the respective process participants (i.e., potential users of the IS to be built) but also other process-specific roles, such as process manager and process owner responsible for the process and its regulation, as well as process-spanning (cross-cutting) roles like process developer (responsible for implementation) and process controller (responsible for process measurement and assessment), or management [17]. Clearly, these roles might have contradicting objectives. For example, a process participant may prefer to have his tasks made easier, while a process owner may prefer to increase process performance.

> → ***Req. 4:*** The prioritization approach should support *the assessment of requirements by different (especially BPM / BPRE) roles.*

> → ***Req. 5:*** The prioritization approach should support requirements being assessed in *different (variable) dimensions* in order to determine their actual business value.

Issue (5): In the context of BPM and BPRE, *objective process performance figures* (e.g., cycle time, number of errors) are often of particular interest [17], as they are more likely to create confidence for decisions than mere subjective assessments.

> → ***Req. 6:*** The prioritization approach should offer the possibility to include *objective value dimensions* for determining the priority of requirements.

Issue (6): As stated above, it is typical for BPRE projects that not only one particular field of duties, but many different processes are of interest for optimization. Thus an *iterative and also incremental* procedure [16] is often performed and recommended, where requirements are elicited successively and the realization of multiple business

processes is executed contemporaneously. This also provides new insights that may change the priority of existing requirements (e.g., a cross-cutting feature that becomes more important because it is needed in different business processes).

→ **Req. 7:** The prioritization approach should support (simple) *re-prioritization and cross-cutting issues* as requirements can change and new ones emerge.

Issue (7): Nowadays, it is common to *adopt business process management or workflow management system*s (BPMS / WFMS) simplifying the implementation of business processes (as opposed to traditional implementation without a workflow engine) [17] in software. As these systems offer a variety of different features that are common to several business processes, the capabilities of these systems already need to be considered during BPRE.

→ **Req. 8:** The prioritization approach should *consider existing features* of the infrastructure, as they may affect the requirements (e.g., lowered costs or risks).

3 Assessing state-of-the-art prioritization approaches for BPRE

In this section, we present an assessment of state-of-the-art prioritization approaches with a special focus on the support for BPRE prioritization requirements as derived in section 2. Table 1 shows the result for the most promising approaches[1], based on an expert assessment of the approaches' description found in the literature.

Table 1. BPRE support of promising prioritization approaches.

Prioritization approach / Requirements	Req 1	Req 2	Req 3	Req 4	Req 5	Req 6	Req 7	Req 8
Cost-Benefit Analysis [8]	-	-	-	o	+	+	-	-
Planning Game [9]	o	-	-	o	-	-	o	-
Hierarchy AHP [4]	-	+	-	o	+	o	-	-
Minimal Spanning Tree Matrix [4]	-	o	-	o	+	o	-	-
Value-oriented HCV [10]	-	+	-	o	+	o	-	-
Cost-value Approach [5]	-	o	-	o	+	o	-	-
Quantitative WinWin [11]	-	+	-	o	-	-	+	-
Moisiadis Framework [12]	o	o	o	o	-	+	-	-
Fuzzy Decision Making [13]	-	-	o	o	+	-	o	-
Interactive GA [14]	-	-	-	-	+	o	o	-
Autom. Requirements Triage [15]	-	o	o	-	-	-	+	o

Legend: - = not supported, o = partly supported or simple adaptation seems possible, + = supported

---

[1] Due to space limitations, we can only show an excerpt of all the approaches we analyzed. Additional approaches that were assessed include: Numeral Assignment, Cumulative Voting, Priority Groups, Top 10 Requirements, Analytic Hierarchy Process (AHP), Bubble Sort, Binary Search Tree, Hierarchical Cumulative Voting (HCV), Wieger's Method, Outranking, Value Oriented Prioritization (VOP), Kano Model, Value Analysis, EVOLVE, Priority Assessment / QFD, Value Based Fuzzy Requirements Prioritization / VIRP, Distributed Collaborative Prioritization, What-if Analysis, B-Tree Prioritization and Binary Priority List.

4   Discussion and solution proposal

As already stated by Herrmann and Daneva [1], most prioritization methods support only one particular part of a larger prioritization process. Even though no detailed discussion of each analyzed approach is possible here due to space restrictions, our analysis has shown that none of the regarded approaches can support all requirements stated in section 2. This is not a surprising result, as most approaches are only designed to solve general requirements prioritization problems. Thus, the selection and application of a specific approach for a particular project strongly depends on the application domain and the prioritization problem at hand. The prioritization problem in BPRE is complex and thus solving it is not possible by applying any existing approach without modifications. *Req. 1* is practically not supported by any of the approaches since they require a final set of requirements and can thus only be applied after requirements elicitation. Fortunately, several approaches support hierarchical prioritization (*Req. 2*) out of the box. Not surprisingly, *Req. 3* is also not supported widely as most approaches only differentiate between requirements on different levels of abstraction (e.g., high level and low level), but do not consider special information available on the different levels (e.g., the number of actors in a use case, as in the work of Moisiadis [12]). *Req. 4* seems to be supported best by available approaches, but only for multiple stakeholders in general, not specialized for different (BPRE) roles. *Req. 5* and *Req. 6* are also supported by several approaches, although some need to be adapted to fulfill these requirements completely. *Req. 7,* in turn, is almost not supported again. Here, sophisticated approaches like Quantitative WinWin [11] explicitly include re-prioritization; however, in most of the other cases, it is hard to do re-prioritization without extensions. Finally, *Req. 8* is not supported by any investigated approach, only Lauren et al [15] mention that relations to existing features "could also be considered".

These results suggest that a prioritization framework is needed for supporting all requirements rather than a single prioritization approach. Based on this insight, we recognize some necessary building blocks to be integrated in such a framework. These are: (1) *a set of prioritization approaches* that satisfy the need for hierarchical prioritization while considering existing COTS features (referring to Req. 2, Req 8.); (2) a *conceptual issue model* that contains the issues (considered in different requirement types) relevant in BPRE, their relationships among each other, and issue-specific information relevant for prioritization (Req. 3); (3) *a value model* that consists of objective (measured) and subjective (assessed by stakeholders) attributes / criteria needed to assess the different requirement artifacts appropriately (Req. 5, Req. 6); (4) a *role model* that describes the roles relevant for prioritizing the different requirements artifacts (Req. 4);   (5) *tool support*, especially for facilitating re-prioritization (Req. 7), and (6) the *BPRE* context in which the proposed framework will be applied for guiding elicitation (Req. 1).

5   Summary

Requirements engineers and decision makers in business-process-driven software

development face the challenge of having to decide which requirements are actually relevant for early business success and should be considered first during elicitation and analysis activities. Our analysis shows that existing prioritization approaches do not support all requirements of typical BPRE projects. In future work, we will further elaborate on the building blocks we have proposed to provide an appropriate solution.

References

1. Herrmann, A., Daneva, M.: Requirements Prioritization Based on Benefit and Cost Prediction: An Agenda for Future Research. In: Proc. of RE '08, pp. 125-134. IEEE Computer Society (Sep. 2008)
2. Riegel, N., Adam, S., Uenalan, O.: Integrating Prioritization into Business Process-driven Requirements Engineering. In: REFSQ '10 - Workshop Proceedings, Essen (2010)
3. De la Vara, J. S., Díaz, J. S.: Business process-driven requirements engineering: a goal-based approach. In: Proc. of BPMDS '07, Trondheim (2007)
4. Karlsson, J., Wohlin, C., Regnell, B.: An evaluation of methods for prioritizing software requirements. In: Information and Software Technology 39, pp. 939-947. (1998)
5. Karlsson J., Ryan, K.: A Cost-Value Approach for Prioritizing Requirements. In: IEEE Software 14, vol. 5, pp. 67-74. (1997)
6. Cardoso, E.C.S., Almeida, J.P.A., Guizzardi, G.: Requirements Engineering Based on Business Process Models: A Case Study. In: Proc. of 13th EDOCW, pp. 320-327. (2009)
7. Salinesi, C., Kornyshova, E.: Choosing a Prioritization Method – Case of IS Security Improvement. In: CAISE ´06 Forum Proceedings, (2006)
8. Nas, T.F.: Cost-Benefit Analysis: Theory and Application. Thousand Oaks, Sage (1996)
9. Beck, K.: Extreme programming explained. Addison-Wesley, Upper Saddle River (2000)
10. Mohamed, A.S.I., El-Maddah, B.I.A., Wahba, C.A.M.: Criteria-Based Requirements Prioritization for Software Product Management. In: Proc. of SERP '08, pp. 587-593. (2008)
11. 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)
12. Moisiadis, F.: The fundamentals of prioritizing requirements. In: Proc. of Systems Engineering, Test & Evaluation Conference. (2002)
13. Gaur, V., Soni, A.: An Integrated Approach to Prioritize Requirements using Fuzzy Decision Making. In: Int. Journal of Eng. and Technology, vol. 2, no.4, pp. 320-328. (2010)
14. Tonella, P., Susi, A., Palma, F.: Using Interactive GA for Requirements Prioritization. In: Proc. of 2nd Int. Symposium on Search Based Software Engineering, pp. 57- 66. (2010)
15. Laurent, P., Cleland-Huang, J., Duan, C.: Towards Automated Requirements Triage. In: Proc. of RE ´07, pp. 131-140. (2007)
16. Cohn, M.: Agile Estimating and Planning. Pearson, Upper Saddle River, NJ (2006)
17. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer, Berlin (2007)

6 International Workshop on Software Product Management (IWSPM)

Editors

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

Marjo Kauppinen
Aalto University, Finland, marjo.kauppinen@tkk.fi

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

Workshop Programme

6th International Workshop on Software Product Management (IWSPM 2012)

Richard Berntsson Svensson[1], Marjo Kauppinen[2], Inge van de Weerd[3]

[1]Lund University, Sweden, richard.berntsson_svensson@cs.lth.se
[2]Aalto University, Finland, marjo.kauppinen@tkk.fi
[3]VU University Amsterdam, the Netherlands, i.vande.weerd@vu.nl

Abstract. 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. Software product management includes working with requirements, release definitions, product lifecycles, the creation and interpretation of product strategies, balancing long-term technology push with shorter-term market-pull, and assuring a successful business case by selecting the right requirement for realization. The 6th International Workshop on Software Product Management (IWSPM 2012) was held at the 18th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'12) in Essen, Germany. The workshop included an invited talk from senior product managers, paper presentations, and discussion on state of knowledge of software product management.

Keywords: software product management; release planning; product release; building practices; product release cycle.

1 Introduction

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.

Software product management includes working with requirements, release definitions, product lifecycles, the creation and interpretation of product strategies, balancing long-term technology push with shorter-term market-pull, and assuring a successful business case by selecting the right requirement for realization.

2 Workshop Goals and Themes

After the success of previous workshops the 6th International Workshop on Software Product Management (IWSPM 2012) aimed at brining practitioners and research experts together for exchanging ideas, knowledge and experience, and at setting a

research agenda based on industry needs. More specifically, IWSPM 2012 pursued the following goals:

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

The theme of IWSPM 2012 was kept in line with the theme of IWSPM 2011, and the themes of interest for paper submission included, but not limited to:

- Product management for software, software-intensive systems, information technology, software as a service, and cloud computing.
- Product portfolios and life-cycles: managing software product innovation based on open source components while providing balanced contribution vs. differentiation strategies.
- Product management in growing size and complexity, ensuring the intended software product qualities with balance between control and innovation.
- Product management for rapidly changing software products, methods for managing the quantity and paste of changes in the most demanding software markets.
- Product planning: product visioning, strategy, roadmapping, and release definition.
- Collaboration in software ecosystems and supply networks: subcontracting, partnering, tendering, negotiation, coordination, and control.
- Business aspects: business case development, business planning, and marketing.
- Product management environments: SME's, large-scale organizations, cross-company ecosystems, and global settings.
- Product management performance: measurement, development, and improvement of product management processes, practices, skills, and competence.
- Tools for product management: innovative SPM tooling, tool evaluation, tools in industry.

Each paper received reviews by three different members of the Program Committee. Program Committee members did not review papers from authors where they had a conflict of interest, i.e. where a Program Committee member and an author came from the same organization or have co-authored papers. Thirteen papers were submitted and from these submissions, four full research papers and two short papers were accepted.

3   Workshop Program

The IWSPM 2012 workshop program covered sessions that shed light on the software product management discipline. The presented papers are included in the IWSPM 2012 proceedings.

Blijleven, Andalibi, Pap and Brinkkemper carried out a study to the influence of internationalization on SPM. They found that no cookie-cutter approaches exist when

it comes to performing product management in international markets, but that there are several factors that are important to consider, such as centrally stored requirements that are accessible to all relevant stakeholders and well-defined and structured communication channels to overcome geographical distances.

During the workshop, two new approaches for release planning were presented. First, Regnell proposed in his work the InnoReap model that can be used to analyze different release planning strategies, in which a trade-off can be made between feature innovativeness and other innovation-related feature properties, such as effort, resource allocation span, and revenue. Secondly, Didar-Al-Alam, Zhi and Ruhe presented EVOLVEII, a release planning approach that considers the impact of technological change by measuring the revised effort needed to perform develop-ment activities.

Van Angeren, van Bommel, Arupia and Brinkkemper investigated the the use and acceptance of bundling as a pricing tool within the software industry. They found that the majority of the companies use bundling as a pricing, delivery and marketing mechanism, which an average of five components per bundle.

Gietema and Brinkkemper carried out a literature review and case studies to find of project management in the product release cycle. This resulted in a list of 10 factors, some related to project management, such as keeping a strict planning, and others more related to the development of product releases, such as managing product interdependencies.

Finally, Zorn-Pauli, Paech and Wittkopf presented an overview challenges for strategic release planning of global information systems gathered from an industrial company in the health care domain and how these challenges are covered in academic literature. They found that many of these challenges are currently not addresses in existing work, and that further research is necessary.

4   Acknowledgements

Software Product Management and Agility

Gerald Heller[1], Hans-Bernd Kittlaus[2]

[1] Software.Process.Management
Tannenstraße 8, 71126 Gäufelden, Germany
gerald.heller@swpm.de

[2]InnoTivum Consulting
Im Sand 86, 53619 Rheinbreitbach, Germany
hbk@innotivum.com

Abstract:

Agile software development has been established in the last 10 years as a popular development approach. In a time when speed of change is of utmost importance, agile approaches are often the most appropriate roads to success. They do not only change the way Development works, but they also impact other parties involved in projects, in particular the software product manager. Software companies are faced with the question how software product management and agile development can work together in an optimal way. Who is responsible for requirements? Is the software product manager automatically the designated "product owner" (Scrum)? Or is "product owner" a new and separate role? Does he/she replace the software product manager?

The Software Product Management Framework which has been developed by the „*International Software Product Management Association*" (www.ispma.org) provides orientation. It can be used as a helpful tool to make the change process towards agile product management successful.

Gerald Heller is Principal Consultant of Software.Process.Management (www.swpm.de). Er has more than 20 years of experience in global software product development. His focus is on requirements and test management in iterative incremental development processes. As a practitioner he is experienced in applying development methods with application lifecycle tools. Gerald Heller has published in the trade press and frequently gives presentations on international conferences. He is Diplom-Informatiker and founding member of ISPMA, and member of IEEE, GI and ASQF.

Hans-Bernd Kittlaus is the owner and CEO of InnoTivum Consulting (www.innotivum.com) and works as consultant, interim manager and trainer for software organizations in the areas of software product management, SaaS, IT strategy and business process management. Before he was Director of SIZ GmbH, Bonn, Germany (German Savings Banks Organization) and Head of Software Product Management and Development units of IBM. He has published numerous articles and books, a.o. "Software Product Management and Pricing – Key Success Factors for All Software Organizations", Springer, 2009. He is Diplom-Informatiker and certified PRINCE2 Practitioner, founding board member of ISPMA (International Software Product Management Association), and member of ACM and GI.

Strategic Release Planning Challenges for Global Information Systems – A Position Paper

Gabriele Zorn-Pauli[1], Barbara Paech[1] and Jens Wittkopf[2]

[1] University of Heidelberg, Im Neuenheimer Feld 326, 69120 Heidelberg, Germany
{zorn-pauli, paech}@informatik.uni-heidelberg.de
[2] Roche Diagnostics GmbH, Sandhofer Strasse 116, 68305 Mannheim, Germany
jens.wittkopf@roche.com

Abstract. In global companies there is a shift from local to global information systems that need to satisfy the needs of many different divisions all over the world. This raises particular problems for strategic release planning, as the succession of releases needs to satisfy multiple business strategies of several countries. Identification of large-scale business aspect similarities, and thus synergies between these strategies, is a strong contributor to success. Features are a common way to represent early requirements or requirement bundles during strategic release planning. Planning global features requires a particular process regarding capturing and selection validation. The goal of this paper is to present challenges for strategic release planning of global information systems gathered from an industrial company in the health care domain. A preliminary literature review investigates to what extent these challenges are already recognized or solved in academia.

Keywords: strategic release planning, product roadmapping, long-term feature selection, global information systems

1 Introduction

The development of information systems (IS) for global companies is changing from locally towards globally oriented customer-specific development, which is reflected by the transition from locally to globally used IS. Globally used IS (abbreviated to global IS in the following) means that due to the globalization of companies, products, and markets the IS needs to satisfy country specific needs of a geographically distributed company. The different company country sites follow to some extent the same global company strategy, but in addition apply for different business strategies depending on country specific settings such as markets, competitors, or regulatory aspects.

Therefore, the integration of multiple business strategies into one global IS imposes major challenges for strategic release planning (SRP). Furthermore, SRP of global IS aims at finding the largest common overlap of multiple business strategies comprising an optimal set of features regarding costs and available resources. For that, important decisions are necessary: *Which features are useful*

*or necessary for most of the company sites and should become a standard func-
tionality in the global IS? Which of them concern only locally driven needs and
should be handled separately?*

Accordingly, global companies need a standardized global IS that still pro-
vides the possibility of locally driven customizations. Therefore, a corresponding
SRP process for global IS is required.

The motivation for companies to shift from local to global IS is primarily
based on organizational aspects such as efficiency enhancement and improvement
of support. Global IS support *(1) global usage of applications, (2) elimination of
inconsistent data* resulting from redundant systems, e.g. when several systems in
different countries support the same processes and *(3) interoperability of business
across different business segments and countries* by cross application and global
master data management.

Within this paper we present challenges for strategic release planning of
global IS gathered from an industrial company in the health care domain.

This paper is organized as follows: Section 2 provides background information
regarding strategic release planning, Section 3 describes the industrial context
and the identified challenges. Section 4 discusses related work and Section 5
concludes the paper.

2 Strategic Release Planning

SRP, also called product or release roadmapping [7] aims at long-term feature
assignment to subsequent releases fulfilling technical, resource, risk and budget
constraints. In contrast, operational release planning focuses only on the devel-
opment of the next software release [11]. The output of the SRP process is a
roadmap document that comprises the future planned features for the software
product and is used for communication and risk or budget estimations. Features
represent the information technology (IT) view of high-level business require-
ments derived from business topics. Due to the long-term planning of SRP the
business needs are not specified in detail and therefore the feature specifications
either. As a result, SRP has to cope with two crucial issues: *(a) fuzzy feature
specifications*, where implementation risks and effort are difficult to estimate
and *(b) continuous re-planning needs*, because of the persistent requests of the
customer for new features or the revision of existing ones.

3 Strategic Release Planning Challenges in Industry

In this section the difficulties for SRP of global IS in the context of a specific
company are explored.

3.1 Global SRP in the Health Care Domain: An Example Company

The company under consideration is active in the health care domain operating
globally in 56 countries. Its global IS is developed by an in-house IT department

and comprises an evolving customer relationship management (CRM) system with country specific local implementations. The CRM system stakeholders are segmented in different company business units such as sales, marketing or service units. Altogether that constitutes a heterogeneous group of stakeholders, which have different business unit priorities. CRM system roadmaps are created per business unit by so called *Change Advisory Boards (CABs)* where the board members involve IT people and business unit representatives comprising the respective key stakeholder in the different countries. Priorities of the specific business units are defined by a company panel and depend on the governance structure. Still, these priorities are not static and can change due to different reasons such as changes in the market or the need to integrate acquired companies.

The elements of a typical roadmap are high-level features, which represent the IT view on the according business topics (e.g. the topic *interoperability of business across different countries* results in a *master data management feature*) associated with a time frame and cost estimations. These high-level features are derived from two different channels. The first channel is business strategy driven based on changing markets, regulatory law or new technology capabilities. The second channel is end user feedback driven where the end users of the IS raise bug, feature or change requests. These requests encompass a pool of requirements of different abstraction levels and are used by IT to suggest further features. Therefore, feature creation is done top-down by refining business topics into features and bottom-up by bundling related low-level requirements into features.

Strategic release planning considers a time horizon of three years that comprises typically two release cycles per year. The focus of SRP activities is on new features neglecting the validation of existing features in terms of usage and suitability.

Since local impacts on a global IS for health care business are very strong, the company aims at providing transnational IS which are oriented on regions such as Asia Pacific and Japan. These regional solutions cluster countries based on geographic distribution and similar market environments. Customization based on regions is assuming that countries, sharing similar markets, also share similar customization needs. At this point *software product line* [10] concepts seem to be appropriate, but there are several reasons why software product line development is not possible or difficult in this company. One reason is that the existing software architecture is not suitable. Another reason is that the company wishes to limit the IS variability and not to encourage it.

3.2 Identified Challenges of Global Strategic Release Planning

The following challenges regarding SRP have been identified together with the health care company and are discussed in this section.

The major problem of SRP for a global IS, based on the authors experiences in the health care domain, is to *balance standardization and customization possibilities* of the IS. On the one side standardization of the IS reduces costs for

planning, implementation and maintenance, but decreases stakeholder satisfaction, since only the business topics common to all stakeholders are considered. On the other side, there is still a need to be able to customize the IS due to country specific needs. In particular, this entails the following four challenges, which may be also common to other domains.

(C1) Identification of Business Strategy Similarities. So far different company country sites have their own local solution without taking advantage of synergies. Examples for such synergies are large-scale reuse similar to product line concepts [9] or identification of business topics that are addressed by many countries and therefore of high priority. So far, the company has managed to integrate multiple business strategies of a small number of countries, by small adaptations of the processes used for local systems. However, since business is an inconsistent environment, the comparison and linking of multiple business strategies are difficult and complex. Thus, for many different countries more powerful methods are needed to support decisions during the strategic release planning and re-planning process for global IS to achieve an applicable combination of customization and standardization capabilities.

(C2) Common Understanding of Global Features. Using global features for release planning requires that several countries must have a common understanding of the features and their relation to the countries own business strategy. Furthermore, during global SRP and alignment with a huge number of heterogeneous stakeholder groups the business topics, mostly represented as features, have to be organized and linked more business oriented. Therefore, the challenge is to utilize business topics for feature creation to get a closer link between business strategies and planned IS.

(C3) Continuous Validation of Roadmaps against Multiple Business Strategies. A roadmap is a living document reflecting the continuous change of business and IS aspects over time. This requires a continuous validation process of the roadmap elements such as selected features against business objectives. A close link between business strategies and planned IS (see C2) is necessary to validate a roadmap against the strategy. Clearly, for multiple business strategies the validation task gets more complex and difficult, as the number of changes is multiple. For example, it is difficult to decide what the right frequency for roadmap validation is or which events call for a re-validation.

(C4) Missing Hybrid Role: Business Engineer vs. Software Product Manager. Planning and developing global IS is a difficult and complex task that requires both deep knowledge about business aspects (e.g. strategies or processes) and technology aspects (e.g. possible mobile data and application access). It is important to have one role responsible for this global SRP. In particular, the

required role would be responsible for the development of new business strategies or models triggered through new IS capabilities or business environment changes (e.g. globalization of markets). This entails that IT takes over business responsibility, which is not always desired by the business. Therefore, a hybrid role, comprising both business and IT power, could encourage the next steps to harmonize business development and according IS evolution.

4   Related Work

In literature there are several approaches and models regarding the SRP process, see [13]. However, all of these approaches neglect the global context of system usage. Suomalainen et al. [12] provide a common product roadmapping process and identified roadmapping process stakeholder. The described SRP process aims at standardized products without considering customization opportunities. [1] introduces a productization process that describes the transition from developing customer-specific software to a standard software product. However, e.g. C1 (business strategy similarity detection) is not supported or considered. Several approaches focus on the enhanced linkage of the business view to the IT view that is part of C2 by aligning business objectives with requirements [6][2][3][5]. Nevertheless, the aspect of global requirements or multiple country business objectives is missing. Integration of variability-based feature modeling during release planning is provided by [4] using feature trees to structure requirements. However, a linking of the features to business objectives for validation of business objective fulfillment (validation according to C3) is not addressed. Related to software product management there exists the role of the product manager which is responsible for creating and maintaining the release roadmaps [8]. It is not clear which additional responsibilities are necessary to fulfill the missing role described in C4.

5   Conclusion

This position paper presented the challenges for SRP of global IS from an industrial perspective. The major problem is balancing standardization and customization possibilities of the IS. For this problem four challenges were identified in a company in the health care domain. A preliminary literature review showed that the problems of global SRP are not addressed in research. It is the aim of our future work to define and evaluate a method for global SRP.

References

1. Artz, P., Weerd, I., Brinkkemper, S.: Productization: Transforming from developing customer-specific software to product software, Lecture Notes in Business Information Processing, vol. 51, pp. 90–102. Springer Berlin Heidelberg (2010)

2. Aslam, K., Khurum, M.: A Model for Early Requirements Triage and Selection Utilizing Product Strategies. In: 14th Asia-Pacific Software Engineering Conference (APSEC '07). pp. 97–104. IEEE Computer Society, Nagoya, Japan (2007)
3. Aurum, A., Wohlin, C.: Aligning requirements with business objectives: A framework for requirements engineering decisions. In: Requirements Engineering Decision Support Workshop (REDECS'05). Paris, France (Sept 2005)
4. Fricker, S., Schumacher, S.: Variability-based release planning. In: Regnell, B., Weerd, I., Troyer, O. (eds.) 2nd International Conference on Software Business (ICSOB'11). Lecture Notes in Business Information Processing, vol. 80, pp. 181–186. Springer, Berlin, Heidelberg (2011)
5. Kauppinen, M., Savolainen, J., Lehtola, L., Komssi, M., Tohonen, H., Davis, A.: From Feature Development to Customer Value Creation. In: 17th IEEE International Requirements Engineering Conference (RE'09). pp. 275–280. IEEE (2009)
6. Khurum, M., Gorschek, T.: A method for alignment evaluation of product strategies among stakeholders (MASS) in software intensive product development. Journal of Software Maintenance and Evolution: Research and Practice 23(7), 494–516 (2011)
7. 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 (RE'05). pp. 439–443. IEEE (2005)
8. Maglyas, A., Nikula, U., Smolander, K.: What do we know about software product management? - a systematic mapping study. In: 5th International Workshop on Software Product Management (IWSPM'11). pp. 26–35. IEEE (2011)
9. de Moraes, M., de Almeida, E., Romero, S.: A systematic review on software product lines scoping. In: 6th Experimental Software Engineering Latin American Workshop (ESELAW'09). pp. 63–72 (2009)
10. Pohl, K., Böckle, G., Van Der Linden, F.: Software product line engineering: foundations, principles, and techniques. Springer-Verlag New York Inc, Secaucus, NJ (2005)
11. Ruhe, G.: Product Release Planning: Methods, Tools and Applications. CRC Press, Boca Raton (2010)
12. Suomalainen, T., Salo, O., Abrahamsson, P., Similä, J.: Software product roadmapping in a volatile business environment. Journal of Systems and Software 84(6), 958–975 (2011)
13. Svahnberg, M., Gorschek, T., Feldt, R., Torkar, R., Saleem, S.B., Shafique, M.U.: A systematic review on strategic release planning models. Journal of Information and Software Technology 52(3), 237–248 (2010)

The Influence of Internationalization on Software Product Management

Vincent Blijleven, Farhad Andalibi, Albert Pap, and Sjaak Brinkkemper

Department of Information and Computing Sciences, Utrecht University
Princetonplein 5, 3508 TB Utrecht, The Netherlands
{v.b.blijleven,f.andalibi,a.p.pap,s.brinkkemper}@uu.nl

Abstract. The role of software product management within firms specialized in product software is of strategic importance, albeit complex to execute. When looking at the role of product management in an international context then, the aforementioned level of complexity tends only to increase. In this paper we present the results of two case studies conducted with software firms that already successfully entered and established themselves in international markets, addressing experienced challenges, issues, and notable differences between conducting product management activities in domestic and international markets. An overview of recommendations based on this research can support other software firms willing to make the step towards internationalization. If product managers take these recommendations into account, better informed decisions could be made and potential pitfalls avoided, leading to higher rates of success and progress when entering international markets.

Keywords: Internationalization, Software Product Management, Software Business, Requirements Management, Release Planning, Product Planning, Portfolio Management

1 Introduction

Internationalization is often seen as a logical next step in the life cycle of a software firm [10]. Plenty of opportunities arise as economic and political barriers fall, as global trade is more and more accepted and modern technology makes it possible to get within reach of a larger customer base. Although much research has already been conducted on the subject of internationalization focused primarily on internationalization strategies and opportunities [4,6,11,13], little research has been conducted on the influence of internationalization on software product management activities. We regard software product management as "the discipline and business process governing a product from its inception to the market or customer delivery and service in order to generate the largest possible value to a business" [7]. Various activities of software product management include for instance requirements management, release planning, product planning and portfolio management.

For firms specialized in product software, the role of product manager is of strategic importance, albeit complex to execute [17]. As solely offering a core

product or service in the software industry is often regarded as insufficient, complementary products and services are required to fulfill the needs of customers [12]. A software product manager is able to identify these specific needs, and can provide relevant information to internal and external stakeholders to address these needs. When looking at the role of product manager in an international context then, the aforementioned level of complexity tends only to increase, for instance due to foreign customer habits [2], lack of legitimacy and influence [9], and a lack of marketing capabilities [1]. Managing the activities of software product management in an international environment in a successful way is thus difficult but crucial in order to sustain, thrive and survive in international markets. A comparison is made of how product management activities are conducted in the domestic market compared to international markets. Based on this comparison, an overview is created containing challenges and issues as experienced by software product managers of firms that successfully established themselves on international markets. This overview can serve as a practical guide for other software firms willing to make this step in order to make better informed decisions and avoid potential pitfalls. The main research question of this paper is therefore as follows; "What influence does internationalization have on software product management activities and deliverables?"

The remainder of this paper continues with a description of the research method in section two, in which we will elaborate on the research methods we employed concerning the case studies and expert reviews. In section three, we discuss the initially identified factors that could be subject to the influence internationalization has on software product management. In section four, we give a short introduction of each firm that participated in the case studies, including the results of these case studies. An analysis of these results presented in section four will be discussed in section five, including tables summarizing the differences when conducting business internationally, the experienced challenges, issues and valuable lessons learned by product managers, and the contribution of this research to software firms willing to make the step toward international markets. In section six, we will discuss encountered validity threats to this research and make statements about generalization possibilities of the results. In addition, we also draw the most important conclusions of this research and provide suggestions for additional research.

2 Research Method

To be able to answer the research question, we made use of two case studies conducted at Dutch product software firms that successfully entered, penetrated and established themselves in international markets. We chose for a multiple case study design in order to get different, unique perspectives on the experienced challenges and issues by software product managers operating in an international context [18]. Because of the exploratory nature of this research, we opted for a structure that is in many ways similar to a recommended case study reporting structure as described by Runeson & Höst [14].

2.1 Case Study Selection Criteria and Validity

The data collection process took place by means of case studies conducted with three software product managers of two medium-to-large commercial product software firms. Two product managers stationed in the domestic market were interviewed, as well as one product manager stationed overseas. The main prerequisite when looking for a potential case study candidate, was that the candidate organization already successfully entered and established itself on an international market, thus having the experience available to provide constructive and meaningful information regarding the experienced challenges, issues and notable differences in the way product management activities are conducted domestically versus internationally.

It should be noted that due to the exploratory nature of this research, the qualitative semi-structured nature of the interviews and the small amount of case studies conducted, generalizability of the results is limited [18]. In addition, unique organizational characteristics of the companies studied such as different organizational structures, different international markets in which the companies operate, and different product-specific characteristics directly influence the generalizability of statements.

2.2 Data Collection and Evaluation of Results

First, an interview protocol was created based on the software product management competence model by Bekkers et al. to serve as a guideline during the case studies, resulting in semi-structured interviews lasting around one hour each [3]. This means the questions were planned and ordered, but not necessarily asked in the exact same order as listed [14]. This protocol was evaluated by means of an expert review with a practitioner specialized in product management in SMEs. Second, three interviews were then conducted with software product managers in order to gather relevant qualitative data on factors of influence concerning internationalization. This lead to the creation of an overview based on the gathered information from these interviews. The interviews were recorded and notes were made during the interviews. Third, the resulting overview based on the qualitative data gathered from the three case study interviews was then evaluated by means of an expert review with an academic specialized in internationalization. We consider the opinion of an academic adequate for assessing real-world situations, as such an evaluation is not biased or lacking judgment from the industry. After this expert review took place, the overview was made definitive.

3 Software Product Management Competence Model

The software product management competence model by Bekkers et al. was the main source of information upon which the semi-interview protocol is based [3]. In this competence model, four main business functions are defined. Within this research domain, a business function can be described as an amount of closely related processes or operations that are performed by a software product manager

(in a routinized way), in order to obtain a defined set of results, contributing to carrying out a part of the mission of an organization. Four different business functions are described within the competence model, these being: requirements management, release planning, product planning, and portfolio management. Each of the mentioned business functions consist of different focus areas, representing a coherent group of capabilities within a business function. With capabilities we refer to important software product management practices.

The business function of requirements management concerns the ongoing management of requirements outside of releases and consists of three focus areas: requirements gathering, requirements identification, and requirements organizing. Release planning then comprises software product management practices required to successfully create and launch a new release, and consists of six focus areas: requirements prioritization, scope change management, release definition, release definition validation, build validation and launch preparation. The product planning business function refers to the collecting of relevant information for the creation of a roadmap for products, product lines or core assets and consists of three focus areas: roadmap intelligence, product roadmapping and core asset roadmapping. Last but not least, the business function of portfolio management concerns the gathering of strategic information and decision making about the entire product portfolio of an organization, and is made up out of three focus areas: market analysis, product lifecycle management and partnering & contracting. Each focus area in the competence model was taken into account when creating the interview protocol, to make sure any software product management related activity according to the competence model, was covered.

4 Case Studies

Three semi-structured interviews were conducted with three software product managers. The interview began with first asking the product manager relevant introductory questions about the organization they work for, followed by questions based on the business functions and relevant focus areas as defined in the software product management competence model by Bekkers et al. [3].

The case study companies were renamed to Alpha and Beta because of anonymity requests. In addition, two product managers (one working in the Netherlands and one working in the United States) working for Alpha have been separately interviewed. A quick overview of statistical data of the companies studied can be seen in table 1. Subtitles in this section printed in bold refer to the four different business functions from the software product management competence model by Bekkers et al. Words in italics are the software product management focus areas.

4.1 Case Study: Alpha

Requirements Management When looking at the business function of requirements management at Alpha, no difference between markets was found

Table 1. Overview of the case study participants

Statistic	Alpha	Beta
Employees	120	1,800
Location of Headquarters	The Netherlands	The Netherlands
Products	Web content management systems, customer-driven online engagement solutions	(Financial) ERP systems, including HRM, CRM and project management solutions
Countries active in	The Netherlands, United States	The Netherlands and 40 other countries

concerning *requirements gathering*. Direct contact with customers, visiting fairs, being present at expositions, and the usage of an international ticket system where customers can report requests are regarded as the main sources of input when gathering requirements. Automated tools are employed to both *identify and organize requirements*, with no direct difference between the domestic and international market. The requirements are centrally *organized* and can thus be easily placed on the product backlog. This product backlog is part of the employed agile Scrum development philosophy.

Release Planning Every market exerts equal influence when *prioritizing requirements*, and a standardized prioritizing methodology is employed. When *preparing a release definition*, no representatives of international markets are present. When *validating a release definition* as the next step, either internal testing (by Alpha itself) is performed or launching customers are used, although this is done solely in the domestic market. *Scope change management* is centrally organized, as the overall corporate vision of the organization is largely responsible for the way in which the product is heading. When scope changes take place, every internal and external stakeholder is informed of scope changes. When *validating a release build*, no use of pilots is made. The product platform is built in such a way that every customer is on its own specific version of the software, thus updating is more or less standardized, can take place whenever the customer desires, and is expected to go flawless. When *preparing for a launch*, information about an upcoming release is communicated to all internal and external stakeholders. Webinars are frequently used to share this information for each market at the same time, although webinars are scheduled to be held separately for each market due to e.g. different time-zones and languages.

Product Planning As aforementioned, the corporate vision serves as the main source of input when creating a (short-term) *product roadmap*. Next to this, direct input is also acquired from all customers, and preferences of customers are measured in various ways, e.g. during specially organized customer event days

where customers can give their opinion on the roadmaps presented during these events. Since a release cycle of three months is used for each product, employees usually think three months in advance. No differences were found with regard to the way *roadmap intelligence* is gathered in international markets, as every product is reviewed in a standardized manner (and in English, thus accessible for every internal stakeholder throughout the organization). Every market is considered equal in terms of influence on roadmaps.

Portfolio Management Analyst sources such as Gartner or Forrester are considered to be of significant value when performing *market analyses*. This is considered especially important in markets such as the United States where the labor market is characterized by loose policies. Employees of relevant companies often consult the aforementioned analysts, in order to be able to clarify the decision they made for a chosen product. Being present in the lists of these analysts is therefore regarded to be of significant importance. Competitor analyses are conducted in a decentralized way, meaning in every market in which the company is active, and later combined to get a general, international overview. Win-loss analyses are also conducted in every market, and are considered to be a source of valuable knowledge. Knowing why (potential) customers decide to buy or not buy a product directly influences product strategies. Customer preferences of the different markets in which the organization operates and the underlying platform architectures of its products exert influence over the *life cycles of the products*. Ultimately however, it is centrally determined, referring to the influence of the corporate vision. Major changes in products are also centrally managed, although external parties are usually involved as they generally possess relevant knowledge. The proposed changes are then reviewed, discussed, written down and centrally stored, accessible for all relevant internal stakeholders.

The introduction of separate product lines is seen as unfavorable, as the current platform architecture is considered to be highly flexible. Introducing new product lines then is considered to create only more overhead. Different versions of software, specifically oriented at for instance the public and private sector are on offer, but are built on the same platform. Regarding *partnering and contracting*, service level agreements (SLAs) are in place taking the differing time zones into account. Support is offered from the domestic market, but various working shifts are employed to be able to provide support throughout the different time zones. External support centers in different time zones are considered due to the increase in the amount of customers. In addition, support preferences are considered to be relatively homogeneous, except for differences concerning national legislation. This is especially the case when targeting governmental organizations that are bound by different rules, requiring the possession of specialized knowledge by support personnel.

4.2 Case Study: Beta

Requirements Management When looking at the business function of requirements management, no difference was found between markets with regard

to the way in which *requirements gathering* takes place from customers. However, the central marketing department plays a pivotal role concerning requirements gathering by performing market research in different countries and the creation of business cases. Interesting market trends are translated to business needs, followed by the creation of conceptual scenarios. Business needs resulting from market research, business cases and customer requirements are then merged to create functional solution. A network of customers and partners is extensively involved from the inception of a solution to its controlled release, which is supported by working in two-week Scrum sprint cycles. Automated tools are used when *identifying* requirements (e.g. to link requirements with similar functionality together), and incoming requirements as provided by customers are *organized* by storing these in a central repository and get updated when necessary.

Release Planning Markets have different weights with regard to the *prioritization of requirements*. This is primarily due to different localizations in which legal aspects have absolute priority, as customers have obligations imposed by governments with respect to e.g. financial reporting. Since a limited amount of development capacity is available, the allocation of developers to address legal requirements has priority. This allocation is largely based on a localization matrix, consisting of five capability tiers of compliance with legislation of all the countries in which the company operates. Apart from legal aspects, the size of the customer base of a specific localization and the results of conducted cost-benefit analyses are also taken into account when prioritizing requirements.

Input is gathered from all markets when *preparing a release definition*. Every market has an equal amount of influence, and the organization is structured in such a way that every country is involved during this process. The corporate vision as outlined by the board, however, serves as the main guideline. When *validating a release definition* then, business cases are created for each market. This is of significant importance due to differences in national legislation. External parties are involved during the validation of the release definition, which is termed a controlled release. Intensive use is made of controlled releases (pilots) when *validating a build*. Launch impact analyses are also conducted for each market to analyse whether the release and deployment of a new build will go without problems. When the controlled releases then went without problems, the build will be released for all markets at the same time.

Differences can be seen when looking at the *launch preparation* for a new build, specifically with regard to timing and internal reporting. The organization strives to make all information as abstract as possible, to which each market can add its own local flavor. Examples are differences in timing (when reporting to external stakeholders) and deciding what information is relevant or irrelevant for a given market. All internal stakeholders are informed about information concerning a new launch at the same time.

Last but not least, *scope changes* to the corporate product line are centrally managed, and impact analyses are performed for each relevant country to measure the effects of possible scope changes.

Product Planning *Roadmap intelligence* is gathered through consultation with all relevant internal stakeholders to analyse what markets are of potential interest to be entered. Overviews showing current and expected upcoming trends within the industry are also created. Overviews showing the big picture of important developments in terms of technology are considered to be of secondary importance to overviews showing important market trends, since it is found that technology is complementary to market demands.

The products on offer have three releases each year, but this amount is not fixed in case there is a good reason to deviate from the scheduled releases. *Product roadmaps* are in place showing the (short-term) vision for each product, and is primarily centrally coordinated. What is interesting to note, is that product management seen from an organizational structure perspective, is as centralized as possible, whereas marketing is as decentralized as possible. The motivation behind this is to allow the decentralized marketing departments to add their own aforementioned local flavor to the products, which is considered to contribute to becoming a local player as much as possible. Being a local player as an international organization within another market is regarded as the highest achievable status, since no distinguishes can then be made between the organization itself and their local competitors. Local competitors are considered to have a natural advantage due to for instance being able to apply their knowledge of their own market and know the mindsets of customers [5].

All *core assets* of the organization are registered and centrally stored. Make-or-buy analyses are constantly performed, to decide whether a given process can be outsourced or should be performed with the available internal resources. Roadmaps are present showing how the core assets are continuously sustained, upgraded and enhanced.

Portfolio Management *Market analyses* are conducted by both internal and external parties, and each market is intensively monitored. Every market has its own marketing strategy, in order to most optimally approach (potential) customers from the respective markets. Competitor analyses performed in each market indicate what actions need to be undertaken in order to become a local player as much as possible. In addition, information resulting from the monitoring of customer preferences in each market is regarded as the backbone of the corporate strategy, as this greatly influences the direction in which the organization is heading. A win-loss analysis is also performed for each large customer, in order to analyze why a (potential) customer did or did not purchase a product and to discover the motivations behind the choices made that lead to the specific outcome. It is considered valuable to know what customers or prospects think with regard to for instance the price of a product, the relationship intimacy with the organization, the reputation, or functionalities offered.

The *lifecycle* of each product on offer is managed on a central level. Changes in the product are managed in a decentralized manner however. This is primarily due to local differences in legislation. In this case, input is sent from decentralized units to the central organization, after which all the input from the different

markets is combined, resulting in an official determination of the product lifecycle. Multiple product lines are maintained for different market types, primarily to optimally address local customer preferences and in order to allow for flexible adaptation to altered national legislation.

When looking at *partnering & contracting*, service level agreements are in place and are considered to be relatively homogeneous among markets, although with some minor differing details. Support to customers is centrally organized, and support needs are also found to be relatively homogeneous among markets. Due to the differences in legislation in each separate market however, support is kept up-to-date to be able to optimally serve each market. This is regarded as one of the core competences of the organization, contributing to the realization of the preferred image of being a local player.

5   Analysis

We presented the case study results for each business function and their belonging focus areas. In this section, we will go in-depth about the interesting findings per business function, in order to keep the closely related processes (focus areas) together. Tables 2, 3, 4 & 5 present differences in the way product management activities are conducted domestically versus internationally for each aforementioned business function in order to get a better overview.

5.1   Requirements Management

In the results it becomes apparent that little difference exists between conducting requirements management activities in the domestic market and international markets (see table 2). The methods employed by both organizations to *gather, identify and organize requirements* are largely the same, except for Beta giving significant responsibility to the central marketing department. The given reason for this, is the usage of a market-driven solution development framework. This involves close monitoring of markets in order to identify new market trends and customer preferences, leading to product requirements.

In addition, both Alpha and Beta apply the Scrum development philosophy when performing activities related to the *organization of requirements*. As conducting business on an international scale tends to increase the complexity of communication and cooperation, the Scrum philosophy is able to e.g. support stakeholders involved (including customers) by focussing on shortening feedback cycles between these stakeholders, reducing time between customer requests and implementations, all of this order to reduce the overall complexity of interaction between relevant stakeholders at different locations [16].

5.2   Release Planning

Both organizations *prioritize requirements* in a different way. At Alpha, each market has an equal amount of influence in terms of speaking volume. Beta

Table 2. Overview of differences concerning requirements management activities

Requirements Management	Influences of Internationalization
Requirements Gathering	No differences, although central marketing department plays a pivotal role depending on the differing geographical locations and time-zones, cultural preferences, languages, and size of the customer base of localizations.
Requirements Identification	Usage of automated tools for all markets to bridge geographical distances.
Requirements Organizing	Centrally stored, accessible to relevant internal stakeholders.

however takes various factors into account when deciding how much influence each market has. Requirements concerning legislation for each localization have absolute priority when prioritizing requirements. Being able to react rapidly to changes in legislation affecting financial processes is regarded as one of the core competences of the organization. In addition, the influence the different markets have is also based on the size of the customer base of each localization, meaning localizations with more customers are prioritized over those with less customers.

The corporate vision of both organizations is the most influential source when *preparing a release definition*. The organizational structure of Beta however is designed in such a way that every market is able to voice its opinion when gathering input. This is performed because of aforementioned differences in national legislation among markets. As a consequence, *validation of releases* take place in all markets in which Beta is active. Alpha validates its release definition only in its domestic market, since not much product-related differences can be seen between the markets, hence causing only more (e.g. communicative) overhead if testing would take place in other markets.

Both organizations centrally *manage scope changes* as the corporate vision is the primary driving force behind these changes. Another difference can be seen when looking at the way in which *build validation* is conducted. Since the updating process at Alpha is standardized, no use of controlled releases is made to validate the build. Beta however makes intensive use controlled releases as well as external parties, since each localization has its own version of the software and thus requires to be separately tested.

Alpha and Beta both employ different approaches to address different markets when *preparing for launches*. Alpha makes use of separate webinars for each market due to different time-zones and languages. Beta relies on decentralization in international marketing, meaning international operations are delegated to separate markets [8]. Product management is located as central as possible, whereas marketing is decentralized as much as possible. Information sent to international markets is made as abstract as possible by the central marketing department based upon input from product management, after which local flavors can be added by the decentralized marketing departments. This allows the

organization to position itself as a local player in international markets with all its benefits, such as a higher level of awareness leading to increased brand reputation, trust and the development of close relationships with customers [15].

Table 3. Overview of differences concerning release planning activities

Release Planning	Influences of Internationalization
Requirements Prioritization	Influence of each market depends on the size of the customer base of a localization, financial aspects such as revenue, and critical market-specific requirements such as legal aspects.
Release Definition Preparation	Centrally coordinated based on corporate vision, although input is generally gathered from all markets.
Release Definition Validation	Validation is only performed in the domestic market, except for when critical market-specific influences on the product demands for separate validation.
Scope Change Management	Centrally managed due to the influence of the corporate vision.
Build Validation	Depending on the nature of the product, business model, and delivery model.
Launch Preparation	Product management is as centralized as possible, whereas marketing is as decentralized as possible to add local flavors. This depends on geographical locations and time-zones, cultural preferences, languages, and size of the customer base of localizations.

5.3   Product Planning

Both organizations consult each market when *gathering roadmap intelligence.* Alpha standardized the way in which products are reviewed per market and the results are accessible to every internal stakeholder in English, thus reducing communicative barriers. Beta combines intelligence received from each market in order to get a bigger picture of global societal and technological trends. When *creating a product roadmap* however, both organizations use the corporate vision as their main source of input. The creation of product roadmaps at Alpha is done entirely by the central headquarters itself, whereas Beta consults the decentralized marketing departments for information and feedback. Both organizations centrally *store and register their core assets.*

5.4   Portfolio Management

Both organizations gather information from each market when performing *market analyses.* Alpha however places additional emphasis on analyst reports such as reports from Gartner or Forrester. This is considered especially important in markets such as the United States where its labor market is characterized by

Table 4. Overview of differences concerning product planning activities

Product Planning	Influences of Internationalization
Roadmap Intelligence	Every product is reviewed in a standardized manner, accessible for every internal stakeholder. Input from each market is then combined to get a bigger picture of societal and technological trends.
Product Roadmapping	Corporate vision is the main source of input and is thus centrally organized, although decentralized marketing departments also provide valuable input.
Core Asset Roadmapping	Centrally stored, accessible to relevant internal stakeholders.

loose policies, making it relatively easy for employers to fire personnel. Employees of relevant companies often consult the aforementioned analysts, in order to be able to clarify the decision they made for a chosen product. Beta performs market analyses in each market by making use of the decentralized marketing departments. Competitor analyses are also performed in each market to identify opportunities and threats.

The corporate vision is again the main factor of influence when looking at the *management of product lifecycles* performed by each company. However, Alpha takes input from each market into account to make sure planned changes to products do not negatively affect markets other than its domestic market. Beta also actively communicates with other markets about planned changes, to ensure no problems arise due to differences in national legislation.

When looking at *partnering & contracting*, both Alpha and Beta regard support preferences and their service level agreements to be relatively homogeneous among markets. Beta experiences minor differences in support questions because of differences in legislation among markets, but this is not regarded as a problem.

Table 5. Overview of differences concerning portfolio management activities

Portfolio Management	Influences of Internationalization
Market Analysis	Analysts are of significant influence when performing market analyses. Every market has its own marketing strategy, and competitor analyses are performed in each market.
Product Lifecycle Management	Centrally organized due to the influence of the corporate vision. Reports from each market are taken into account, and changes in the product are managed in a decentralized way depending on market-specific circumstances.
Partnering & Contracting	SLAs and support preferences are considered relatively homogeneous among markets.

5.5 Lessons Learned

At the end of the interviews, the case study participants were asked whether they had learned valuable lessons they would like to share. The case study participants indicated **no cookie-cutter approaches exist** when it comes to performing product management activities in international markets. There are no real 'one-size fits all' solutions since for instance every new request, issue, or activity demands another approach. The complexity of product management in foreign markets should thus not be underestimated.

Another lesson learned, was that organizations should **decline customer requests that negatively affect the organization in the long run**. An example given was declining requests that involve tailor-made software for different localizations. Beta experienced that delivering tailor-made software seemed like a lucrative deal at first sight, but became a real burden as several years passed. As the tailor-made software slowly became legacy software and Beta itself as an organization evolved over time, nobody eventually knew who was responsible for the development of what part of the tailor-made software. This resulted in problems concerning support that lead to time-consuming processes and unnecessary costs. Beta now considers the acceptance of tailor-made software proposals as misplaced customer-driven behavior. Organizations should therefore always think of long-term consequences when considering a customer request.

Attempting to overcome cultural differences in other markets is regarded as another underrated problem. Alpha indicated that the organization consistently had the feeling it was in a less favorable position compared to local competitors, due to societal factors such as chauvinism. Becoming a fully integrated local player is considered to be near impossible. This creates **the need for differentiation and the propagation of unique selling points** compared to products offered by competitors.

Alpha also mentioned the problem of experiencing communicative barriers. Due to operating in different time-zones and the relatively long geographical distance between offices, employees working in the United States indicated that they experience problems with being kept up to date. No longer is interesting information for instance exchanged with fellow colleagues during short conversations while standing next to a coffee machine. This implies **the need for well-defined and structured communication channels**.

6   Discussion & Conclusion

We addressed the influence internationalization has on software product management activities and deliverables. This was done by means of conducting case studies at internationally operating product software companies in the form of semi-structured interviews. This semi-structured nature was the result of using an interview protocol based on both a literature study and an expert evaluation. We found that internationalization influences all business functions described in the software product management competence model by Bekkers et al [3].

When looking at the business function of requirements management, we found that decentralized marketing departments play a pivotal role in gathering requirements to most optimally address customers. In addition, requirements are centrally stored and accessible to all relevant internal stakeholders to bridge geographical distances and time-zones. The size of a customer base of a localization, revenue, and critical market-specific requirements such as legal aspects are of significant influence when looking at the business function of release planning. This is especially the case when prioritizing requirements. The preparation and validation of release definitions, management of scope changes and launch preparation is performed as centralized as possible, except when foreign market-specific issues are involved, for instance legal aspects. The costs and effort of performing these activities abroad was found to outweigh the benefits it could offer. The corporate vision defined by the executive board was found to be strong factor of influence when looking at the business functions of product planning and portfolio management. It is the main source of influence concerning product roadmapping and product lifecycle management. In order to get a bigger picture of global market demands and societal and technological changes however, relevant stakeholders in foreign markets are also consulted. SLAs and support preferences are considered to be relatively homogeneous among markets.

Among lessons learned by the interviewed product managers were that no cookie-cutter approaches exist when it comes to performing product management activities in international markets, and that customer requests that negatively affect the organization in the long run should be declined. The need was also felt to differentiate and propagate unique selling points in international markets, in order to compensate for disadvantages international newcomers have compared to already established local players. They also mentioned the need for well-defined and structured communication channels to overcome geographical distances between offices that impose communicative barriers.

It has to be noted that generalizability of the presented findings is limited due to the small amount of case studies conducted. Unique organizational characteristics of the companies studied such as different organizational structures, different international markets in which the companies operate, and different product-specific characteristics also directly influence the generalizability of the findings. Next to this, the organizations studied offer different types of products and services (e.g. a web content management system versus an organization-wide ERP system). This does not make both organizations directly comparable when comparing how specific activities are performed within an international context.

Future research should be directed at gathering a larger and more representative dataset, for instance by conducting a structured survey in the product software industry. Conducting additional case studies will continue to provide useful data, although a larger dataset is preferred in order to make plausible statements when comparing organizations. A larger dataset will also allow for the division of companies in categories, making it possible to make statements about findings between different product type companies. Additional research should also aim to include survey participants that have their headquarters lo-

cated in different countries, to study the specific effects of political-geographical influences on conducting business from another perspective. Last but not least, future research can also orient itself toward researching specific parts of internationalization, such as whether the usage of external partners in international markets positively influences the internationalization trajectory.

References

1. Alajoutsijarvi, K.: Customer relationships and the small software firm a framework for understanding challenges faced in marketing. Information & Management 37(3), 153–159 (2000)
2. Arenius, P.: The psychic distance postulate revised: From market selection to speed of market penetration. Journal of International Entrepreneurship 3, 115–131 (2005)
3. Bekkers, W., Weerd, I.v.d., Spruit, M., Brinkkemper, S.: A framework for process improvement in software product management. In: Proceedings of the 17th European Conference on Systems, Software and Services Process Improvement (EUROSPI), Grenoble, France, 1-12. (2010)
4. Bell, J.: The internationalization of small computer software firms: A further challenge to "stage" theories. European Journal of Marketing 29(8), 60–75 (1995)
5. Bhattacharya, A., Michael, D.: How local companies keep multinationals at bay. Harvard Business Review 86(3), 84–95 (2008)
6. Coviello, N., Munro, H.: Network relationships and the internationalisation process of small software firms. International Business Review 6(4), 361–386 (1997)
7. Ebert, C.: Software product management. CrossTalk 22(1), 15–19 (2009)
8. Jobber, D.: Principles and practice of marketing. McGraw-Hill (1995)
9. Leonidou, L.C.: An analysis of the barriers hindering small business export development. Journal of Small Business Management 42(3), 279–302 (2004)
10. McHugh, P.: Making it Big in Software - A guide to success for software vendors with growth ambitions. Rubic Publishing, England (1999)
11. Moen, O., Gavlen, M., Endresen, I.: Internationalization of small, computer software firms: Entry forms and market selection. European Journal of Marketing 32(9/10), 1236–1251 (2004)
12. Moore, G.: Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers. HarperBusiness (1991)
13. Reuwer, T., Jansen, S., Brinkkemper, S., (n.d.): Key factors in the internationalization process of smes exporting business software as a service. International Journal of Business Information Systems (forthcoming)
14. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14, 131–164 (April 2009)
15. Schuiling, I., Kapferer, J.N.: Executive insights: Real differences between local and international brands: Strategic implications for international marketers. Journal of International Marketing 12(4), 97–112 (2004)
16. Schwaber, K.: Agile Project Management with Scrum. Prentice Hall (2004)
17. Weerd, I.v.d., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L.: Towards a reference framework for software product management. In: Proceedings of the 14th International Requirements Engineering Conference. pp. 319–322. Minneapolis/St. Paul, Minnesota, USA (2006)
18. Yin, R.K.: Case Study Research: Design and Methods. Sage Publications, Inc (2008)

Managing the Product Release Cycle

Ten factors determining success in project management of product release cycles

Suzanne Gietema and Sjaak Brinkkemper

Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands
s.l.e.gietema@uu.nl, s.brinkkemper@uu.nl

Abstract. This paper discusses ten success factors of project management in the product release cycle. Unlike general project management in release management the cycle of the same project is repeated, releasing a new version of a product with every new release. A literature study and case studies executed at several companies have shown that success factors to managing development of product releases can be divided into two domains: management of the release and contents of the release. The ten success factors that were found are: strict planning, clear communication, interim checks, full documentation, realistic customer expectations, feasible and understandable requirements, downsize functionality, performing an impact analysis, managing product interdependencies, and strictly separating releases. Some of these success factors are also key to general project management while others are more specific to managing the development of product releases.

Keywords: Product, release, management, success, factors, cycle

1   Introduction

Each project needs to be managed to make it a success. Project management can be defined as: the scheduling, monitoring and control of a project and the motivation of everyone involved to reach the project goals within the set time and to the specified cost, quality and performance [1]. Project management does not necessarily mean the management of one project cycle, meaning when a project runs from start to finish and then never is revisited. Many IT-projects involve software products that are subdue to many release cycles. Products of which a newer updated version of the product is released over a period of time. Some of these release cycles are constant, where a product update is released e.g. every three months. Other release cycles are slightly more sporadic, where one update may be released three months after the previous one and another update may take six months or even longer.

This research paper looks into the project management of the product release cycle. The basis for this research project is the research question: "What are success factors of project man-

agement in the product release cycle?". The authors of this paper view each product release development as a project, from the initial development of a release to the actual release of the new version of the product. The authors define the product release cycle as the continuous development of new versions of a product. A product experiencing this, continuously moves through the various stages of the development cycle. Key factors of successful project management of products involved in release cycles have been studied through a literature review and case study research. Software product release management is defined by Jansen and Brinkkemper [2] as "the storage, publication, identification and packaging of the elements of a product". Software product management and software project management are interrelated. Manteli, Weerd and Brinkkemper [3] created a Software Management Conceptual Model showing the interrelationship between Software Product Management and Software Product Management, describing three independent variables of project management (time, quality, and cost) in relation to three dependent variables of product management (market success, customer satisfaction, and business goals), and also discussing the distinct roles of product manager and project manager. Research has shown that product managers and project managers need to understand each other's roles to be able to communicate and collaborate efficiently [3]. Project management success involves meeting project goals like time and financial objectives [4]. Product success involves the capability of the final product to meet the project owner's business objectives like customer satisfaction or the satisfaction of stakeholders needs if related to the product [4]. Research has shown that project management success has a positive relationship with product success [5]. This paper focuses on what project management factors make product release development successful, it will not focus on the factors that make an actual product successful.

2 Related literature

Research has shown that there are several key factors that make project management and product development a success. These next sections discuss project management, product management and product releases.

2.1 Project management

A project can be defined as a temporary effort aimed at creating a unique product, service or result. Project management is the use of knowledge, skills, tools and techniques in project activities with the purpose to meet project requirements [1]. Project management entails applying and integrating several project management processes. These processes are initiating, planning, executing, monitoring and controlling, and closing. Part of these processes is the identification of requirements, creating clear and achievable goals, creating a balance between quality, scope, time and cost, and modifying the specifications, plans, and approach to the various concerns and expectations of stakeholders [1].

Cooke-Davies [6] determined project success factors by answering three questions: "What factors are critical to project management success?", "What factors are critical to success on an individual project?", and "What factors lead to consistently successful projects?". It was determined that several factors are key to in-time performance. These factors are: sufficient organization-wide instruction on the principles of risk management, process maturity for assigning

risk ownership, sufficient maintaining of a risk record, a current risk management plan, enough documentation describing organizational responsibilities on the project, and project (stage) length should be kept as far under three years as possible. Key success factors to the on-cost performance of a project are: allow modifications to project scope only through a mature scope modification management process, and retain the integrity of the performance measurement baseline. Cooke-Davies [6] also points out the difference between project success and project management success. Stating that the first is determined by the general project objectives, while the latter is determined on the basis of cost, time, and quality. In this way success criteria for projects and success factors for project management are distinguished from each other. It was determined that a critical factor for project success is an effective benefits delivery and management process involving shared collaboration of project management and line management functions. Atkinson [7] depicts the iron triangle including three interdependent key factors of project management also mentioned by Cooke-Davies [6]: cost, quality and time. Meaning that if one factor decreases or increases the other factors will be also decrease or increase. Other research [8] identifies an additional factor to these three factors, being scope. In this research it is defined that only three of the factors can be fixed, but one will always remain variable.

Clarke [9] poses that by focusing on key success factors, like communication and clear objective and scope, problems related to project management that distract from the main goals of a project can be solved. It is stated that by applying the following key success factors to problems in project management, projects will run better. Communication creates better understanding, helps eliminate mistakes, may increase motivation, can help identify problems sooner, and encourage team-work. As a result the project's chance to meet its goals within the set time and allocated resources increases. The objectives and scope of the project needs to be clear to create general understanding, which will make it more likely that everything needed is included in the project and nothing is being forgotten [9]. Instead of just working with large projects it is better to break them down into "bite sized chunks" making the project easier to handle. This will result in more ownership by those owning a part of the project, because responsibilities and accountabilities are spread amongst more people. Project management is made easier in a number of ways because of this. By creating a better overview of a project part, which is easier to review than an entire project. Also project plans should be considered to be dynamic, because it is unlikely that a project will completely go to plan. Clarke [9] stresses that it is the interaction amongst these success factors that is key to the success of a project not the factors by itself.

It is important to maintain project knowledge [10], especially within projects where knowledge needs to be reused (e.g. for a new release). Important project experiences should be captured immediately, with the whole project team, when achieving a significant milestone. Also individual group member's experiences should collectively and interactively be analyzed. These insights should be used to increase understanding as to what actions hold what consequences.

The Standish Group [11] found that projects that are successful are completed on time and within the set budget, including all the features that were initially specified in the project requirements. They identified ten factors that attribute to project success. These factors are: executive support, user involvement, experienced project manager, clear organizational goals, minimized scope, standard software infrastructure, strong basic requirements, a formal methodology, trustworthy estimates and other criteria (including good planning and capable staff).

Andersen et al. [12] broke down project success factors into parts of different project-stages. This is depicted in Table 1.

Table 1. Project success factors after Andersen et al. [12])

Project Stage	Success factors
Foundation	Align the project with the organization Get commitment of involved managers Create a shared vision
Planning	Use numerous levels Use simple friendly tools Encourage creativity Make realistic estimates
Implementation	Negotiate resource availability Cooperate Define management responsibility Get commitment of resource providers Define channels of communication Project manager should function as manager not chief technologist
Control	Integrate plans and progress reports Formalize the review • Clear intervals • Clear criteria • Controlled attendance Use sources authority

2.2 Product Development

A product can be defined as anything tangible or intangible which can be bought. Businesses make products in order to sell them to other businesses or consumers. A product by itself can be resold to a consumer, or it may be sold as part of another product [13]. A product may be part of a product line, which can be defined as several related products grouped together, focusing on markets that are alike, or on solving a specific type of problem. Products that are part of the same product line serve markets that are alike or can be created through a similar production method. A product line can also be viewed as a small product portfolio. Product portfolios hold groups of related product lines [13]. Fig. 1 depicts the Product Portfolio Structure.
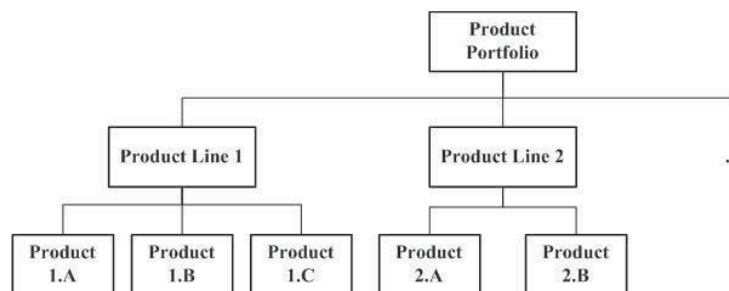
Fig. 1. Product Portfolio Structure (based on: Haines [13])

Product management can be defined as organizational management of the product, product line, or project portfolio level, which is handled by the product manager. A product manager is often responsible for part of or the entire product platform, architecture, a module or set of modules, a single product, a product line, or a product portfolio [13]. Therefore it is important that a product manager focuses on the right success factors and knows what factors of failure are in order to make project management of the product release cycle successful.

The product life cycle consists of several phases [13]. The product life cycle model shows what stages a product goes through from the initial idea up to it being finally managed while in the market. This model is important for product release, because with each new version a new product life cycle is started. The product life cycle, which shows product portfolio work area buckets like feasibility, development and growth, is depicted in Fig. 2. This is overlaid with a cash flow diagram, illustrating that the introduction of a product does not start to pay off until a significant amount of time has gone by.

The way a company handles product development can give it a great competitive advantage [14]. Proper management of a product can greatly influence software product success [15]. Also the quality level of Software Product Management processes may also tremendously influence software product success, due to quality improvements and the prevention of delaying releases [16]. Weerd et al. [16] created a maturity matrix to help assess an organization's Software Product Management capabilities as well as providing incremental enhancements to the product manager. Reducing new product development time can give a company a relative advantage in market share, profit and long-term competitiveness. In this process cost, performance, schedule and quality should also be handled.
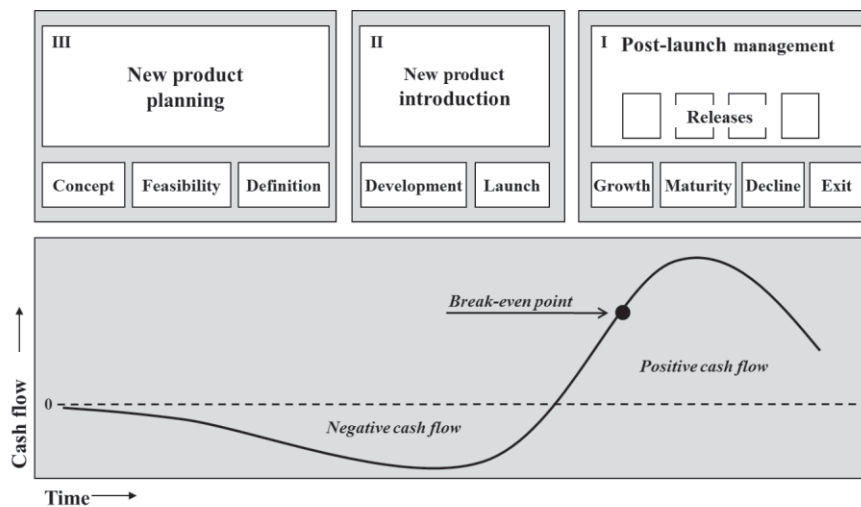
Fig. 2.   Product Life Cycle model (Based on: Haines [13])

Research by Manteli, Weerd and Brinkkemper [3] has indicated that quality is key to market success for a software product. It was also concluded that product managers judge the cost variable for having the greatest influence on meeting business goals. Other research by Janssen

and Brinkkemper [17] have also found that the maintenance of software vendor's customer relationships is essential to being successful and surviving within the current industry climate.

Where product development works with various continuous product releases there is an extra dimension added to the product development process. In the complex release process [18] there is weight on the time-to-market with a strain on the amount of resources available [19]. Requirements associated with each release version have to be carefully planned and managed. Also selecting which requirements to implement in a release is an important part of the release process [20]. Organizations focusing on software development often have successive software product releases, in this process release planning is key [21]. This planning, however, needs to be flexible [22].

Jansen and Brinkkemper [17] identified ten misconceptions about product software releasing. Examples of these misconceptions are that customers want and must stay up-to-date, repairs can be put off until the next major release, workarounds need to be avoided at all costs, the next release is always superior for the customer than the previous release, and that releasing too often is not a good thing. Jansen [23] found four facts indicating that a software knowledge base improves release processes. Management of product data improves the release of regular products. Information collected during product development can be reused in later phases like implementation. Also storing knowledge centrally leads to a reduction in delivery effort. Finally being able to ask "what-if" questions to a local software knowledge base that is linked to several component sources may increase the trustworthiness of the component deployment process, assisting a system manager to better predict which adaptations can be made to a system and what features can be provided with a change.

In the previous sections many factors for product success and project management success have been identified. Due to the special circumstances surrounding product release additional success factors may be expected for products involved in a release cycle, besides the regular success factors associated with project management or product management.

3   Methodology

This research focusses upon finding success factors for project management of products subject to release cycles. To build further upon the success factors identified in theory, the empirical part of this research entailed the performance of case studies have been to identify product release cycle management specific success factors. For this specific research topic the management of the product release cycle a multiple case study for theory building [24] used, to create a theoretical basis for further research. These case studies were performed at a variety of organizations with experience in release management, in the form of an interview.

In these interviews focus was put on how each case study company manages their release cycle. Where major focus points lie in that release cycle and what each company views as important factors for making project management of product release cycles a success. The interviews were performed in a semi structured way, i.e. the interviewer took the lead in asking the relevant questions, but provided room for the interviewees to show and tell what they considered relevant. The interviews therefore were not extensively guided. The data gathered in the case study interviews represents the insight given by the interviewee from his/her perspective.

4 Case study research

Each case study company has its own approach in dealing with product releases, however over the course of the case studies it was noticed that there are some general success factors to product release cycle management. Also it seems that the product release cycle, due to its more unique circumstance of being a repeating project, has many additional success factors as opposed to a regular one time project. Table 2, provides general information about the case study companies and the interviewees. These companies have been given fictitious names, representing the type of product/service they provide.

Table 2. General Company information

Company	Interviewee position	Year founded	Product / Service	Number of Employees	Number main of Products	Duration Release Cycle
RealComp	Director software development	2000	Real-estate software	44	4	4-5 months
WorkComp	Vice President Software Development	1984	Software- and Consultancy services involving Integrated Workplace Management solutions	320	6	4-5 months
ERPComp	Project manager	2009	Advice on process improvement and product development	4	4	9 months
FinComp	Requirements manager	1927	Finances	107000	2	4-5 months

4.1 RealComp

RealComp is a small firm that specializes in real-estate software. According to RealComp key to the product release process is especially managing the start of the process. This is where the process basics are determined. Requirements management is important, often the list of product requirements is too long to be realized when also coping with time and budget. A selection has to made, so the most important requirements remain. Also a time estimate has to be made when determining scope, where the high level estimate versus the actual available capacity. According to RealComp the estimate is often off by a factor two or more. It is important to have a finished conceptual design before the start of the iteration, to have everything clear and prevent problems due to incomplete information. There is a constant (so called) game going on between scope, budget, and timing, with the dilemma of where cuts should be made, weighing between increasing the budget, taking functionality away from the scope, and increasing the time span on the development.

New releases may influence any previous releases as well as other products related to the product of which a new version is released. Therefore differences on where products are in their respective life cycles may influence the product release cycles of products releases currently being developed. The effect a new release has on related products as well as previous releases needs to be taken into account. It should be made clear to the customer what will be developed far in advance of the actual release (at least six months), however make sure to provide a disclaimer that this could be subdue to change. Conceptual documents should always be completed before the actual product development starts, this will speed up the release development. Problems are often caused by last minute changes, due to a lack of design, because product development continuous due to time pressures. Commitments that are made have to be managed, meaning that customer expectations need to be managed. Also the points where business meets technology needs to be managed, meaning can what the business wants be realized within the timespan set with the resources available. A delay on one release may result in parallel tracks with the next release and result in additional work. Instead of delaying a project it might be better to leave out some of the functionality. To overcome this problem time boxing may be used, where some functionalities only make it in if timing allows it.

4.2   WorkComp

WorkComp is a small software company that has two to three releases per year. They define product development as an interdependency of three variables: quality, scope and time (a), using three of the four, in literature, mentioned variables influencing development and a variation of the iron triangle.

According to WorkComp it is important that you don not fixate all three of these variables, because that would leave little room for error. In WorkComp's case quality needs to be sufficient, time is fixed and scope is flexible. Meaning that when a deadline cannot be made some of the functionality might be taken out in order to make the deadline. Depending on the type of functionality it may be put in the next release instead of the current release, if this does not cause too much strain on the next release. WorkComp offers full support for two years on any old release, however they try the phase out old releases as quickly as possible. Since any changes to an old release takes away from the development time of a new release, which may cause pressure on the development process. In development WorkComp uses the waterfall method. This method uses stages where one stage leads to another, starting with a design, then making an estimate, then planning, and then development. Between design and estimation is a high level of uncertainty, meaning that it is uncertain if what was designed can actually be realized within the estimated time. An estimation can vary to 30% of its original estimate according to WorkComp. To manage risks within this entire process good requirements management is key. Requirements may change due to a changing environment, making the product scope uncertain. Keeping these changing requirements in check, will help manage the development process. If time would not be fixed, meaning a release date would not be set, this could cause problems, there will be continuous changes in the product and no usable version of the product will be released. It is important to manage the customer's expectations, to make sure these are realistic and are in line with the product being developed. Product management and product marketing need to be in sink, to make sure that marketing is promoting the release features being created.

4.3   AdviComp

AdviComp provides advice on IT projects, including product releases. According to AdviComp time and cost are secondary functions in product development, while functionality and quality have a primary function. When taking a look at the product life cycle the most errors are made in the business case, at the start of the life cycle where the product is in development. Also project management needs to manage communication between marketing and development, so that what is being developed is also what is being marketed to the customer. Project management is key, because it manages product expectations. Product development is influenced by three variables, similar to the variables depicted in Fig. 3a by WorkComp. These variables are, scope, time and budget (fig. 3b), also using three of the four mentioned variables influencing development and a variation of the iron triangle.

While working on a product release it is important to continuously keep an overview of who is responsible for what. Statuses need to be communicated and updated and communication channels need to be set in order to keep everything clear. Requirements management as well as clear planning is important to the development process. When falling behind on planning it might be decided to take out some functionality, important in this process is timely communication. It does not have to be a problem when releases of the same product are being developed simultaneously, however it does take away from the next release. Deadlines should therefore be maintained as strictly as possible. Clear information management is also important in the development process in order to steer the development process in the right direction. Information needs to be shared and there needs to be a coherence between the departments working together, not stand alone teams, in order to have an efficient and effective product development. A sense of awareness and responsibility needs to be created. This can be done by asking "what do we want to achieve?" and analyzing how this can be achieved (Fig. 4).
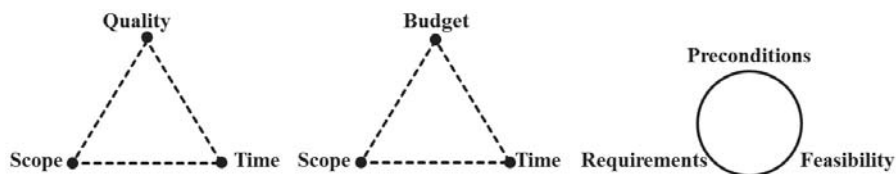
Fig. 3a, 3b, and 3c Interdependency between the three key variables in product release development according to the case study companies
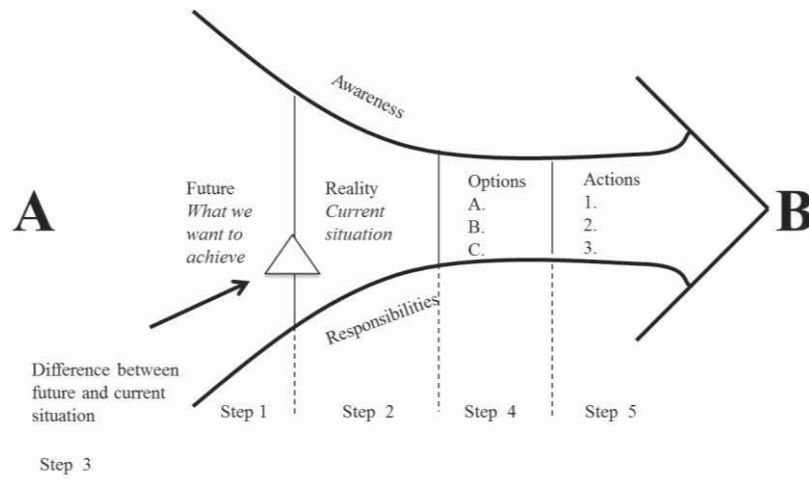
Fig. 4.   Creating awareness

4.4   FinComp

FinComp is a large Dutch financial institution, which has multiple IT projects with several releases per year. Multiple factors are key to successful product release development according to FinComp. A tight planning, specifying deadlines and release dates, is important. This planning should only be filled for 80 percent of the time, leaving room for repairs and unexpected situations. Even though planning maybe filled for 80 percent at the start of a release it can happen that planning does not offer enough time. By making requirements autonomous, it is easier to move a requirement to a next release.

An impact analysis should always be done in order to determine what the feasibility is in relation to what the company wants to achieve. Also a regression test should be done in order to retest everything there already is, to prevent basing anything new on something faulty.
The product development process consists of several stages which are depicted requirements creation, design, build, test, implementation, and after care.

Parallel release cycles do not have to be a problem when releases are in a different stage of their development process. Disciplines shift from one release to the next and each stage requires mostly different disciplines and therefore different people. Management should therefore try to steer towards a situation where releases are developed in parallel. However, it is important to have constant configuration management, which means that releases and their documentation are kept strictly separated to avoid any confusion.

FinComp distinguishes the same product development interdependent variables as AdviComp (figure 4b): time, scope and budget. Whenever there is a change to an existing product, there is a change in requirements. It is important to have quality checks of these requirements as well as checking the preconditions set for the requirements. Those having to work with the requirements afterwards should always be part of this check, to make sure if they understand and are able to work with the requirements as they are. This process is depicted in figure 3c, where the circle depicts the release requirements and around it shown what needs to be checked

for. Everything in a release should always be completely documented so steps can be tracked back and provide clear information to potential new project members.

FinComp states that mistakes are often made at the beginning of the release process as well as in the transition from one stage to the other. Creating good requirements can be hard, since they do not always cover the problem that requires the change. Therefore the problem should always be clearly described, defining what the problem is, where it is occurring and how it can be solved. Communication about the problem is also essential and often goes wrong. Often the what, why and gravity of the problem is depicted wrongly. Also with each stage there should be a check of the previous stage, e.g. does the design match the requirements that were set for it and does the product match the design.

Finally it is also significant to the release process to have the right people do the right work, meaning that people should be working according to their capabilities.

5 Ten success factors of product release cycle management

Judging from the case studies conducted at the various companies, described in the previous chapter, a variety of ways for handling the product release cycle are used. However, ten general success factors were identified for the product release cycle when analyzing these case studies. These success factors can be divided into two domains *management of the release* and *contents of the release*. These factors helped answer the research question *"What are success factors of project management in the product release cycle?"* and are depicted in Table 3. These success factors also include more general project management success factors, like planning and communication, as was described in the literature section of this paper.

Table 3. Success factors of product release

Domain	Success Factors
Management of the release	1. Strict planning, including deadline, checks and contingency reserves. 2. Clear communication amongst project members. 3. Interim checks during the entire process. 4. Full documentation for every release. 5. Realistic customer expectations of the product to be released.
Contents of the release	6. Feasible and understandable requirements specification. 7. Downsize functionality when necessary. 8. Performing an impact analysis. 9. Managing product interdependencies. 10. Strictly separating releases.

SF1: Strict planning, including deadline, checks and contingency reserves

Keeping a strict planning, that includes deadlines and interim checks is key to delivering a release on time, because it gives restrictions as well as provides clarity for project members. By planning out the entire process and having additional contingency reserves for unexpected situations like error fixing, it is more likely that a high quality release will be delivered within the intended time frame.

SF2: Clear communication amongst project members

Communication amongst project members is important within the release process to ensure that what is required is being build and that what is being build is actually necessary. For example if a company would build a new release of their product including a feature that it automatically changes the color every 5 minutes, but there is actually no desire for this new feature, there is no point in building it. Communication helps to determine the what, why and necessity of a release, as well as creating awareness amongst the project members.

SF3: Interim checks during the entire process

During the entire release development process there interim checks should be performed, meaning when moving from one development stage to another it should be checked if everything is understandable and done in the specified manner. For example did the design team design what was specified in the requirements, but also are the requirements specified in such a way that the design team will be able to create a design consistent with the requirements, and is the actual product build in accordance with the specified design. These checks are necessary to prevent major repairs to the created release, which would increase development costs and development time.

SF4: Full documentation for every release

Everything related to a release needs to be documented, because documentation depicts what is done, when something was done, and why something was done. It will help everyone involved understand the release better and current documentation will provide much input for the next release.

SF5: Realistic customer expectations of the product to be released

The customer expectations about the release should be managed to make sure these expectations are realistic. Due to many changes happening quickly in the IT industry changes in the final release may occur as opposed to the original scope of the project, therefore the customer should be made aware of this. Make sure stories about the product release don not grow out of proportion, to make sure the customer does not have any unrealistic views of the product release being created.

SF6: Feasible and understandable requirements specification

Requirements management is key to product release management because requirements are at the basis of product development. They determine what the end result of the product being created/adapted will be. Therefore requirements need to specify the intended change and cover the problem that needs solving. Requirements have to be feasible and understandable, so they have to clearly describe the intended change as well as being realizable within the intended

time. E.g. too many requirements for a release will make realization of the next release within the planned timeframe less feasible.

SF7: Downsize functionality when necessary
In the game between time, scope, money and quality, time and money are often fixed. This means that a company cannot take forever to develop a release, as well as there not being an endless source of money. Therefore functionality should be taken out when time or cost does not allow to develop the entire planned scope or give it the full desired initial quality. Otherwise there will be unnecessary costs, as well as an overrun on time which might also cause problems for sequentially planned releases.

SF8: Performing an impact analysis
Every new release, has an impact on the organization it is developed by. The impact of each release should therefore be checked. This will indicate if the new release covers what the organization wants to achieve, if the organization has the ability to build the release (is it feasible) and also if it is worth spending the required resources to achieve the set goal.

SF9: Managing product interdependencies
Managing product interdependencies is necessary, because any change to a product might affect a related product. Therefore the influence of a new release on the performance of related products should always investigated. A new release is a lot less useable if it prevents any related product from working.

SF10: Strictly separating releases
Often there might be a parallel track of releases of the same product, where one release is still in its early stages, while another may almost be ready to go to market. To avoid chaos and misunderstanding there has to be configuration management, meaning managing that the releases and their documentation are kept strictly separate from each other. In this way preventing situations where functionalities of a new release are specified in an earlier release, or the other way around.

Keep in mind that it is the combination of the success factors that makes project management of release cycles a success and that more of one factor will not necessarily mean an organization will me more successful in managing their release cycle. As with most things, too much is often also not a good thing.

6 Conclusion and further research

This study has shown that there are several ways for handling the management of the product release cycle. This paper has provided an overview for several companies working with IT product releases and what they view as key success factors to this process. It was determined that ten factors are key to product release cycle success. These factors can be divided into two domains, *management of the release* and *contents of the release*. The management factors focus on planning, communication, interim checks, documentation and customer expectations. The content factors focus on requirements, functionality, impact, product interdependencies, and the

separation of releases. Several of these key success factors, like communication and planning, are also key success factors for project management in general.

This research looked into how project management of the product release cycle can be made successful, however it did not look into how product release can be successful in accordance with these factors. As well as what the influence is of lacking one or more of these success factors is. Further research should look into what the gravity is of each success factor in relation to developing a product release. Also it further research should look into the influence of these success factors on the success of a product release.

6   References

[1] Project Management Institute, A guide to the Project Management Body of Knowledge, 3 ed., Newton Square, Pennsylvania, USA: Project Management Institute, 2004.

[2] S. Jansen and S. Brinkkemper, "Ten Misconceptions about Product Software Release Management explained using Update Cost/Value Functions," in *First International Workshop on Software Product Management*, 2006.

[3] C. Manteli, I. v. d. Weerd and S. Brinkkemper, "Bridging the gap between software product management and software project management," in *Proceedings of the 11th International Conference on Product Focused Software*, New York, 2010.

[4] D. Baccarini, "The logical framework method for defining project success," *Project Management Journal,* vol. 30, no. 4, pp. 25-32, 1999.

[5] A. Collins and D. Baccarini, "Project success - a survey," *Journal of Construction Research,* vol. 5, no. 2, pp. 211-231, 2004.

[6] T. Cooke-Davies, "The "real" success factors on projects," *International Journal of Project Management,* vol. 20, no. 3, pp. 185-190, 2002.

[7] R. Atkinson, "Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other criteria," *International Journal of Project Management,* vol. 17, no. 6, pp. 337-342, 1999.

[8] K. Beck, Extreme programming Explained, Addison-Wesley, 2000.

[9] A. Clarke, "A practical use of key success factors to improve the effectiveness of project management," *International Journal of Project Management,* vol. 17, no. 3, pp. 139-145, 1999.

[10] M. Schindler and M. Eppler, "Harvesting project knowledge: a review of project learning methods and success factors," *International Journal of Project Management,* vol. 21, no. 3, pp. 219-228, 2003.

[11] The Standish Group International, "Chaos report," The Standish Group International, 2009.

[12] E. Andersen, K. Grude and T. Haug, Goal Directed Project Management, London: Kogan Page, 2003.

[13] S. Haines, The Product Manager's Desk Reference, USA: McGrawHill, 2009.

[14] P. Afonso, M. Nunes, A. Paisana and A. Braga, "The influence of time-to-market and

target costing in the new product development success," *International Journal of Production Economics,* vol. 115, no. 2, pp. 559-568, 2008.

[15] C. Ebert, "the impacts of software product management," *Journal of Systems and Software,* vol. 80, no. 6, pp. 850-861, 2007.

[16] I. v. d. Weerd, W. Bekkers and S. Brinkkemper, "Developing a Maturity Matrix for Software Product Management," *Lecture Notes in Business Information Processing,* vol. 51, no. 1&3, pp. 76-89, 2010.

[17] S. Jansen and S. Brinkkemper, "Evaluating the release, delivery, and deployment processes of eight large product software vendors applying the customer configuration update model," in *Proceedings of the 2006 international workshop on Workshop on interdisciplinary software engineering research*, Shanghai, China, 2006.

[18] B. Regnell and S. Brinkkemper, "Market-driven requirements engineering for software products," in *Engineering and managing software requirements*, Dordrecht, Springer, 2005, p. 287–308.

[19] R. Novorita and G. Grube, "Benefits of structured requirements methods for market-based enterprises," in *International council on systems engineering sixth annual international symposium on systems engineering: practice and tools*, 1996.

[20] B. Regnell, L. Karlsson and M. Höst, "An Analytical Model for Requirements Selection Quality Evaluation in Product Software Development," in *11th IEEE International Conference on Requirements Engineering*, Monterey Bay, California, USA, 2003.

[21] P. Carlshamre and B. Regnell, "Requirements lifecycle management and release planning in market-driven requirements engineering processes," in *IEEE International. Workshop on the Requirements Engineering Process*, Greenwich, UK, 2000.

[22] S. Sawyer, "A market-bases perspective on information systems development," *Comminications of the ACM,* vol. 47, no. 12, pp. 97-102, 2004.

[23] S. Jansen, "Alleviating the release and deployment effort of product software by explicitly managing component knowledge," in *Proceedings of the Workshop on Development and Deployment of Product Software*, 2005.

[24] J. Dul and T. Hak, Case Study Methodology in Business, Elsevier Ltd., 2007.

Software Release Planning Incorporating Technological Change – The Case of Considering Software Inspections

S. M. Didar-Al-Alam, Junji Zhi, Günther Ruhe

Software Engineering Decision Support Laboratory
University of Calgary, Calgary, AB, Canada
{smdalam, zhij, ruhe}@ucalgary.ca

Abstract. Release planning is a cognitively and computationally complex task. It assigns features to different releases considering technological, business objectives and constraints. Current planning techniques ignore the impact of technological changes. However, these changes are more the rule than the exception.

Our proposed approach considers the impact of technological change. Our model measures this impact in the revised effort needed to perform development activities. While scope of technological change might potentially be very broad, we focused on introducing software inspections, a technique empirically proven to increase software development effectiveness.

In this paper, (i) a theoretical method of quantitatively incorporating technology change impact into existing release planning model is discussed.; (ii) a solution method using EVOVLE II is proposed; and (iii) results of an illustrative case study using an example new technique software inspection are analyzed. For the case study, real-world planning data and data about the impact of inspections taken from extensive range of empirical studies were used.

Keywords: Release Planning, Technology Adoption, Product Management, Case Study, Software Inspections

1   Introduction

Release planning decides which feature to assign in which release [1]. Software companies strive to make good release plan to attract customers and to satisfy stakeholders. The ultimate goal of a good release plan is to achieve competitiveness and general business success. On the other hand, in the context of fast-changing technology era, software companies are facing the need to employ new technology into their product lines or development process. Such industrial innovation and ability to confront the emerging technology is important in terms of a company's success because it can help companies sustain a competitive advantage [2]. Existing research shows that computer software industry needs to improve their work in carrying out transition to new technology [3]. By effectively and coherently introducing new technology or new tools, software companies can achieve the goal of shortening development lifecycle, and, meanwhile, improving the quality of products or services. Research has shown

that some organizations, especially those who are subject to the increasing product market competition, show a higher propensity to adopt technological innovations to improve their products or services [4]. Usually adopting new technologies does not start from scratch. Most companies have a deployed product line and owned product road-maps. One of the important concerns for managers is creating product release plans by adopting new technology considering its product line and product road-maps.

Current release planning models often consider technology as an element as unchanged during the development cycle. What happens to the old plans when the technology as an underlying factor has changed? The systematic integration of technological change into existing release planning model has not been fully studied. This is the main problem area analyzed and discussed in this paper. In this paper, (1) a theoretical method of quantitatively incorporating technology change impact into existing release planning model is discussed, (2) critical information needed for applying these approaches are stated; and (3) results of illustrative case study using an example new technique software inspection are analyzed.

Some terms need to be clarified before stating the problem statement. Information technology (IT) is defined as the acquisition, processing, storage and dissemination of vocal, pictorial, textual and numerical information [5]. Technology adoption (TA) is considered as studies regarding technology from a sociological perspective. In this paper we considered introducing a new technology in software companies. The employees are software practitioners, playing both roles as technology adopters and implementers. Most studies of TA are based on sociological models but the results of these studies can be applied to software companies also but only with appropriate modifications. The illustrative case study is an example of introducing inspections on a real world release planning scenario. The paper is structured as follows. Section 2 describes related works of the problem context. Section 3 briefly describes about release planning. Section 4 explains our methodology used in this study. Section 5 exhibits the setup of our illustrative case study and Section 6 analyzed the results obtained from the case study, our achievements and limitations and Section 7 provides the conclusion and future research possibilities.

2   Related Work

Incremental software development focuses on smaller releases of software products sequentially with time, instead of waiting for a long period of development. It is crucial to choose among the features for the earlier releases of the software product [6]. Considering technical precedence among the features, the stakeholders' choice and balance between required and available effort, all features are prioritized [6]. Release planning determines a collection of features for future releases that is most attractive to the users and the customers [1]. Some popular approaches of feature prioritization are studied and compared by J. Karlsson et.al. in [7] which are useful in release planning. These methods includes AHP [8], binary search tree (BST) creation, greedy approach etc. G. Ruhe et. al. in [9] presented a quantitative study of different software

release planning schemes under risk and resource constraints. Our paper has considered EVOLVE II for discussing software release planning. EVOLVE II is a systematic method for planning product releases [1] that introduces hybrid intelligence, a combination of human and computer intelligence for software release planning [10].

Technology adoption (TA) brings ready technology to the users. TA is an interdisciplinary research study, involving economics, psychology, management, engineering, technology itself, etc. [4, 11–13]. M. Huggett in his paper [13] shows that adopting new technology may initially lead to productivity fall and then later rise. Five broad factors like, commitment, knowledge, communication, planning and infrastructure are identified in [12] which are associated with information system implementation success or failures. Researchers examined how and when one organization accepted technology in their decision-making efforts in [14]. S.A. Brown et. al. in [15] reveals a different pattern of relationships in mandated use of new technology situation compared with voluntary technology adoption cases where ease of use and perceived usefulness are mentioned as the primary and the second determinant of employees' behavioral intention.

Technology road mapping (TRM) is a useful visualization technique to support decision-making, strategic planning and to enhance communication. TRM decision-making process is complex, involving different aspects such as technology, management and business. Some researchers propose T-plan process to support starting-up process of the firms [16]. B. Yoon et.al. proposed a methodology for TRM in [17] where morphology analysis (MA) derives technological opportunities and assist TRM. M. V. Zelkowitz et. al. presented in [3] a discussion about how industrial organizations evaluate new technologies. A. W. Brown et al. proposed a theoretical framework to evaluate a new technology in [11] which can serve as a basis for a systematic approach for research community to evaluate software technologies. Features are essential abstractions [5] that both customers and developers understand. EVOLVE II plans for product releases [1]. It integrates the involvements of all major stakeholders and provides decision support for them using hybrid intelligence [10].

Software inspections (SI) are widely believed to be the most cost-effective method for detecting defects. Researchers described the scope for inspection tool support and review currently available products in [18]. [19] provides quantitative evidence for the benefit and cost of SI and also points out the reluctance of project managers to apply SI. C. Jones et.al. evaluates different methods of training software professional personnel in the context of 2008 financial meltdown and rapidly evolving new technologies [20]. Authors presented the importance of training in an organization especially for a new tool or technique in [21]. A course curriculum related to software inspection training is presented in [20]. Empirical study on software employees training and its impact is presented in [22]. Training need analysis for an organization depending on training budget has been analyzed in [23].

3   Release Planning

Software product features are essential abstractions that both customers and developers understand [5]. Release plan is assignment of features to certain releases. To achieve competitiveness and general business success, software companies are concerned about how their release plans can attract customers and get higher satisfaction. Making good release plans requires the decision-makers to assign attractive product features to the right release under the constraint of limited resources. Resource here refers to all human and non-human resources which are utilized to implement the features. All features have specific requirements for resources. But total resource capacity is limited and lower than the total demand of features. Release planning objective is to choose the best combination of features for a release which are possible to implement within the limited resources. Stakeholders provide prioritization of their feature demands which helps to predict which features are most beneficiary to offer in a future release.

Release planning problem can be modeled as a collection of N features F = {f(1), f(2),…,f(N)} that need to be decided when to release. The term 'feature' is literally broadened and can refer to new functionalities, change requests or defect corrections [2]. The solution to the problem is then represented by decision vector x = (x(1),x(2),…x(N)), when

(1)   x(n)=k if feature is offered at release k (n=1… N)

(2)   x(n)=0 if feature is not offered in any release.

We consider T types of resources for the implementation of features. In its simplest case, for T = 1, we just consider the total effort needed. Further, we define Cap(k,t) as the capacity of the tth type of resource in the kth release. Each feature demands an amount r(i,t) of resources of type t. Therefore, the problem- is to come up with a solution vector x = (x(1),x(2),…x(n)), which satisfies the constraints:

(3) $\sum_{x(i)=k} r(i,t) \leq Cap\,(k,t)$

for all releases k and all resources type t and maximize the objective function. The objective function can be different depending on the release planning framework utilized. Further details about release planning methods and tools can be found in [1].

4   Methodology

The key hypothesis of this paper is that, technological changes have a substantial impact on the structure and content of release plans. The impact of employing a new technology can be substantial in a software organization. For the purpose of simplicity, we let the impact be reflected in the feature implementation effort estimation and resource allocation. Technological change problem is modeled numerically as the resource capacity re-distribution and implementation of effort re-estimation along with addition of extra efforts.

4.1   Release Planning without Consideration of Technological Changes

Different release planning models are proposed to address problems encountered in different scenarios. One of the systematic methods for planning product releases is EVOLVE II [1]. EVOLVE II method integrates a number of new concepts and their innovative implementation, since the initial publication of EVOLVE in [9]. EVOLVE II relies on the paradigm of hybrid intelligence [10] and follows the broadly applicable idea of evolutionary problem solving [7]. The interaction between formalized problem solving based on the application of specialized optimization algorithms and the capabilities of human experts. The steps and their content are listed in Table 1. For the initial case of the case study we consider a release planning procedure using EVOLVE II framework without considering any technological changes.

Table 1. Steps of EVOLVE II.

Step 1:	Specification of key parameters of the planning problem based on corporate strategy and the specific project information available.
Step 2:	Determine weights of relative importance of criteria.
Step 3:	Stakeholders with pre-selection role prioritize all features in order to select a reasonable set of candidate features for the more comprehensive, multi-criteria prioritization and subsequent planning. This step includes a validation of the pre-selected features.
Step 4:	Prioritization of features by stakeholders following the multi-score method is performed for all defined criteria.
Step 5:	It gives an overview of the ranking of features and the degree of commonality in stakeholder opinions.
Step 6:	Stakeholders with assigned resource-estimation role analyze the set of candidate features and available resource types. They estimate the amount of resources consumed by each feature.
Step 7:	The product manager formalizes the features' technological (dependencies, pre-assignments) as well as all other resource or risk related constraints.
Step 8:	Optimization algorithms are applied in order to obtain a portfolio of five optimized and diversified release plan alternatives.
Step 9:	Analysis of the optimized alternatives in terms of their quality and resource consumption.
Step 10:	Projection of the expected reaction of the stakeholder in terms of excitements (with the assignment of individual features), disappointments, and surprises related to the assignment of all the individual features.

Step 11:	The question answered by the what-if-analysis is how certain changes in the problem space (the specific project data) imply changes in the solution space.
Step 12:	Selected stakeholders are asked for their priorities related to the proposed plan alternatives.
Step 13:	Final plan selection based on the previous evaluation and analysis reports.

4.2 Incorporating Technology Change Factors

Numerous factors associated with technology change can be identified only by working jointly with a real life project. A new technology adoption model for an organization is specified in [14]. TA decision is done under a number of technical and philosophical factors like the family of the new tool, comfort of use, compatibility with previous tools and business demand etc. These factors affect some considerable criteria like *perceived ease of use* and *perceived usefulness* values in adopting new technology. The benefit and risks related to the new technology along with the overall adoption process of a new technology gets affected from all these considerations. Moreover, some important parameters to be considered for TA are like - how the software is developed, which life cycle model is used in development, tools and technology that are applied, team size, parameters that the company uses to describe its development, effort allocation, the degree of usage of the new tool, the training need etc. In our illustrative case study the scenario has been simplified by limiting our scope to the major affecting factors only like - effort allocation, degree of usage of the new technique, and training. All other parameters are considered out of scope of our study for the time being and used in our study with constant values. We also narrowed down the scope for the term "new technology" for simplicity by exclusively considering the impact of inspections as an example for looking at the impact of technological change on software release planning. In our study we considered release planning using the Releaseplanner[TM] tool [24] and EVOLVE II release planning framework.

4.3 Problem formulation

Release planning framework like EVOLVE II considers an effort matrix M_e shown in equation (4) below where the value of $r_{[i,j]}$ represents the effort cost for the ith task in j^{th} feature f(j) It is considered that if N is the maximum number of features and T is the maximum number of resources available then equation (4) is true for the ith task in j^{th} feature f(j) if $(0 < i \leq T)$ and $(0 < j \leq N)$. In our model, we considered technology change factors impacts effort estimation. The using new technology during the overall feature implementation process is considered as a new task for each feature. In order to reflect this idea in a formalized way we added a new row

$$(4) \qquad M_e = \begin{pmatrix} r_{[1,1]} & r_{[1,2]} & \cdots & r_{[1,N]} \\ r_{[2,1]} & r_{[2,2]} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ r_{[T,1]} & r_{[T,2]} & \cdots & r_{[T,N]} \end{pmatrix}$$

$v = (r'_{[T+1,1]}, r'_{[T+1,2]}, \ldots \ldots, r'_{[T+1,N]})^T$ in the effort matrix M_e. This newly added row represents the new task efforts for using the new technology. Besides, the earlier effort cost $r_{[i,j]}$ value is also re-estimated and represented by $r'_{[i,j]}$. Re-estimation is done because employing new technology impacts the resource consumption of each feature. The modified effort matrix is visible in equation (5) below.

$$(5) \qquad M'_e = \begin{pmatrix} r'_{[1,1]} & r'_{[1,2]} & \cdots & r'_{[1,N]} \\ r'_{[2,1]} & r'_{[2,2]} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ r'_{[T,1]} & r'_{[T,2]} & \cdots & r'_{[T,N]} \\ r'_{[T+1,1]} & r'_{[T+1,2]} & \cdots & r'_{[T+1,N]} \end{pmatrix}$$

Along with these changes resource capacities are also re-estimated. Let $Cap'(k, m)$ represent the capacity of the mth type of resource in the kth release for $m = 1 \ldots (T + 1)$. Re-estimated effort values are dependent on the inspection scheme used to re-estimate effort and resources. Re-estimated effort matrix M'_e is visible in equation (5). M'_e satisfies the resource capacity constraint shown in equation (6), for all releases k and all resources type m.

$$(6) \qquad \sum_{x(i)=k} r'(i,m) \leq Cap'(k,m) \qquad \text{for } m = 1 \ldots (T+1)$$

Generally, the total resource capacity for each release remains equal. The new task added for adopting new technology also consumes resource from this limited capacity. The assumption is based on the fact that software organizations usually do not change resources, before or after introducing a new technology. This assumption expressed in a formal way in equation (7) for all releases k where T is the total number of resources.

$$(7) \qquad \sum_{i=1}^{T} Cap(k,i) = \sum_{i=1}^{T+1} Cap'(k,i)$$

4.4 Formulation of the Effort Estimation Model

Illustrative case study assumed a software company plans to employ SI to improve developers productivity and quality of the software products to reach a higher level in software maturity. SI helps software developers to avoid predictable pitfalls and through static testing verifies that software meets its requirements [21], [25]. SI is a proven technique to improve quality and reduce costs [26]. For illustrative purpose in our case study, results from Perspective Based Reading (PBR) [27] inspections are analyzed. It costs approximately additional 10% of the total feature implementation effort to complete the inspection.

The systematic integration of technological change into existing release planning model requires a smooth technology transition from current plans towards the new technology. Training is a crucial issue in it. Training means acquiring new skills,

knowledge regarding a specific task through proper organized form of teaching in order to improve productivity and performance. Without proper training provided, introducing new technology in organization is highly risky and time intense [22]. Training also introduces important change factors to the organization like- adds additional effort [21], consumes resources [21], and employees productivity is highly dependent on the provided training [22]. Training need is dependent on numerous factors. For simplification we limit our scope of study only to the major factors like learnability and knowledge level of participating employees [22].

Training also has a final impact on the total effort change through affecting employee's productivity. Training depends on the learnability & knowledge level of the participants. N. Hanakawa et. al. presented a knowledge model in [22] which clearly illustrates, how staff's knowledge level and learnability changes with time and productivity changes with this learning and experience gathering process.

Table 2. Relationship between Productivity and Inspection in different releases.

Release	Knowledge	Productivity	Effort spent in releases for PBR (% of total effort)
Release 1	Training for basic knowledge	50% [22]	20%
Release 2	Low improvement	65%	15
Release 3	High improvement	95%	11%
Release 4	Experienced	100% [22]	10%

In our case study we assume that staff's productivity reaches from initial 50% of their maximum productivity [22] to their maximum 100% productivity within four releases. The growth of productivity from release 1 to 2, 3 and 4 is shown in Table 2. Productivity estimation in Table 2 follows the productivity pattern suggested in [22].

5 Illustrative Case Study - The Case of Introducing Software Inspections

The illustrative case study is performed using our own developed simulation application and the ReleasePlanner[TM] [24] tool that implements the EVOLVE II framework. Experimental data is based on one of our industrial partners' previous project with some minor changes. All assumptions are considered with valid proof of justifications. The basic scenario does not include any technology change and compared with projects that incorporated technological changes. Main attributes of basic scenario are shown in Table 3. We assumed different scenarios where inspection and training efforts are applied. Our simulation application modifies the basic project data according to flexible choice of inspection schemes, inspection team size and training schemes etc. ReleasePlanner[TM] tool creates release plans for this new project. Results are compared against the basic scenario.

Illustrative case study considered PBR inspection scheme. Training for technology adoption is a crucial issue. We calculated from [20] that maximum of 24% of total

employees can be involved in inspection. In illustrated case study for total employee of 25 persons we considered the team size for inspection is 5 person [20]. Software inspection training topics and time and different schemes are presented in [20]. The illustrative case study considered some known training programs and their time limits [20]. We have also considered training effort changes over release to releases [22]. In our application user can flexibly choose the value for these parameters to see its impact on technology change.

Table 3. Major project settings in basic project without inspection

Type	Value	Type	Value	Weight
No of features	100	No of releases	4	
No of feature groups	6	Release weights	Release 1	9
No of stakeholders	91		Release 2	7
No of total employee	25		Release 3	6
Inspection team size	5		Release 4	5
No of resources	5	Planning criteria	Sales	9
No of tasks per feature	5		Quality	9

Table 4 represents the effort for training and inspection estimations for the project when PBR inspection was first introduced (PBR1) and a follow up project (PBR2). Employee's productivity is different between releases while PBR is first introduced. As the employee gets more experienced the productivity increases over time. Assumed staffs productivity are also visible in Table 4.

Table 4. Productivity and Training value for projects with PBR Inspection

Description	Release 1		Release 2		Release 3		Release 4	
	PBR1	PBR2	PBR1	PBR2	PBR1	PBR2	PBR1	PBR2
Inspection (% of total effort)	20%	10%	15.3%	10%	10.5%	10%	10%	10%
Training (days)	62.5	5	22.5	5	10	5	5	5

6 Analysis and Discussion of Results

6.1 Overview

The illustrative case study considers a scenario of a software company which is planning to introduce a new technology and already have a well-established release plan. Major focus of our study is on the changes happened in the organization due to changes in the technology. The impact of technology change will finally be reflected by effort changes. Our estimation model shows these changes based on empirical data

obtained in earlier research works. Our model is based on the study of [25] that can lead us to the venn-diagram shown in Figure 1. According to this PBR eliminates approximately 23% rework effort while adds additional 10% of the total feature implementation effort to complete the inspection.

Analyzing these changes in different possible scenarios helps us to understand how introducing a new technique can affects the current release plan of an organization. Results obtained in this case study are utilized to discuss the main goal of this paper. Does technology change have an impact on release planning that need to be considered? We limited our discussion in three different scenarios which includes a

1) project without any inspection (baseline case)
2) project where inspection is first introduced (called PBR1) and
3) follow up project of earlier projects where inspection was introduced earlier (called PBR2).

In rest of our discussion we will denote these three scenarios as S1, S2 and S3 correspondingly.
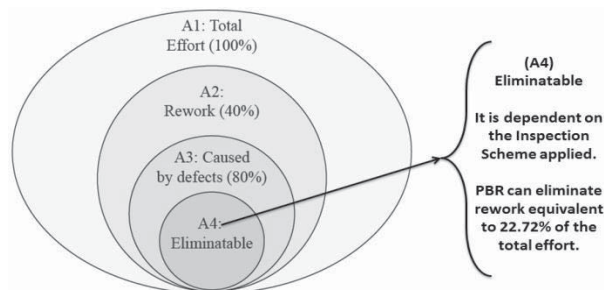

Figure 1. Rework elimination for inspection (Data adopted from [25])

6.2 Incorporating Technology Change Factors in Release Planning Framework:

Adopting a new technology like SI introduces additional task with extra effort consumption in existing release planning scenario. Throughout our study we used EVOLVE II [1], a well-known framework for strategic decision making, to plan software releases. We considered additional effort for inspection as a new task "Inspection" for each feature. Training is considered as features, to be completed by a group of employees with different level of efforts in different releases. Considering the impact as a change in effort distribution makes it easy to incorporate in release planning framework. As the total resource capacity is fixed the leftover resources from rework elimination are utilized to serve SI tasks and Training features. SI effort differs from release to release depending on the productivity and training needed by the employees. Generally in later releases the training need gradually minimizes and productivity grows higher [22].

6.3   Changes in Selecting Features between Different Scenario Release Plans due to Adopting New Technology

A comparative study must be done in between all the scenarios considered in the case study to measure the changes taken effect in the release plans due to the adoption of new technology. Hamming distance measures the number of changes between two equivalent elements like two strings. So we considered using Hamming distance as metric to measure the number of changes in feature selection between different scenarios projects release plans. For three different scenarios we considered three pairs of comparison. The comparison is done for four releases separately and for the features that are been postponed in all four releases. The ratio of feature change correspondent to the total features offered in a release is considered in calculation. To normalize the results, the number of change in feature selection is presented as a percentage in respect of the total features offered by the releases.

The results of the comparisons are shown in Figure 2. Our focus is not to show any specific pattern in the changes between releases or between project scenarios. Instead, the focus is to point out this major phenomenon that adopting a new technology has a high impact on the release plan. Most of the criteria are considered constant but the effort distribution for each scenario is different due to the adoption of the new technology. This has caused such highly visible differences among the release plans. From above discussion and Figure 2 it is clear that technology change is a highly potential issue to be considered while planning for future releases to make it more adaptable to changes.
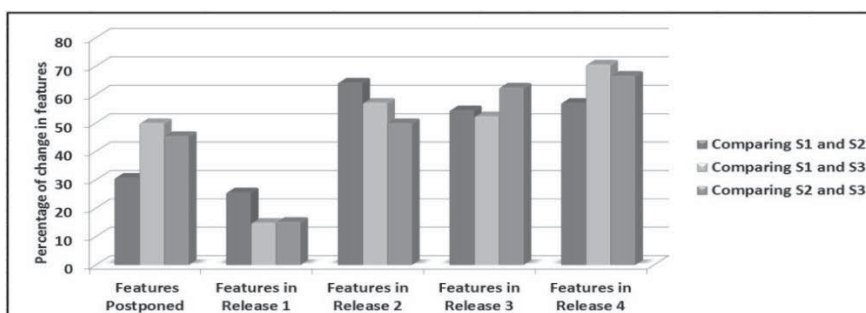

Figure 2. Comparison of feature changes in release plan for different scenarios

6.4   Changes in the Quality of Release Plans

To analyze the quality of release plans, of our case study the major parameters considered are- total stakeholder feature point (TSFP) and stakeholder feature point (SFP) [1]. A clear conceptual view of SFP, feature scores and TSFP is presented in [1]. These two parameters reflect the satisfaction level of the stakeholders.

Figure 3(a) presents a comparison among project scenarios S1, S2, and S3 in respect of their TSFP values. TSFP is depicted on the Y axis and the different scenarios

are depicted on the X axis. Introducing a new technology creates a clear difference in the release plan performance in respect of TSFP which is clear in Figure 3(a).

(a)   (b)
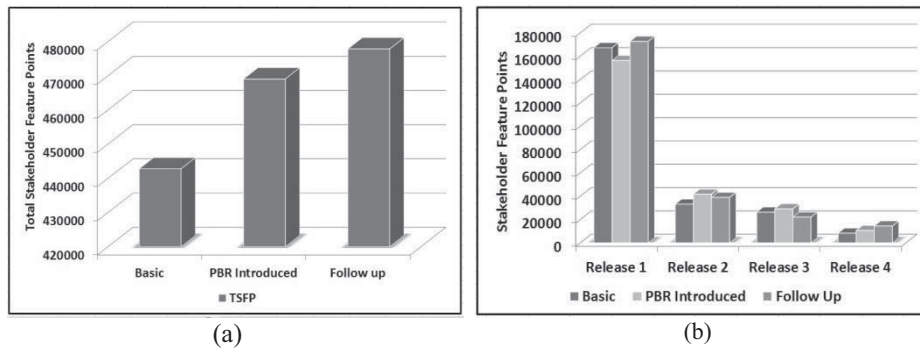
Figure 3. Comparison among the projects using (a) total stakeholder feature points (TSFP) and (b) stakeholder feature points (SFP)

Well managed technology adaptation resulted in TSFP increase as shown in Figure 3(a). This increase of the TSFP value directly corresponds to an increase in customer satisfaction. A more detailed view is available in Figure 3(b). Therein, SFP are plotted for the different releases and the three scenarios under scrutiny. Some degrade in quality happened because of low productivity of untrained employees in scenario S2 when the inspection is introduced for the first time. But in the follow up project training requirement is minimal and the productivity is maximum as PBR is already implemented. It shows benefit in Release 1 and 4 in this our example over the past two scenarios in respect of SFP.
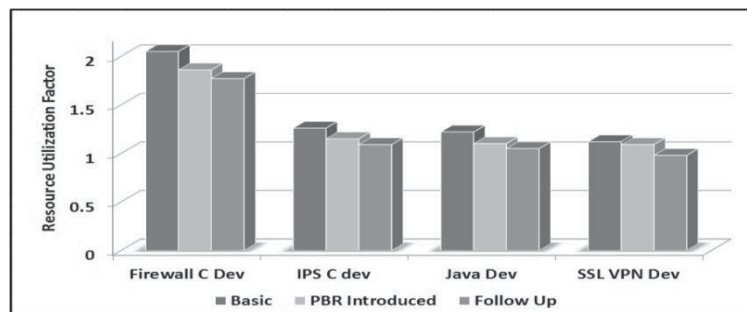
Figure 4. Comparison among the scenarios using Resource utilization factor

Resource utilization factor (RUF) is the ratio of total resource usage in respect of the total available resource. RUF represents the efficiency of a release in respect of resource usage. Lower RUF value means, less difference between available resource and used resource and higher resource utilization. In Figure 4, we considered RUF value to compare results among the scenarios, to understand how well they utilize their resources. RUF value is presented in Y axis in respect of different tasks in X axis. Different scenarios are presented in different shades. RUF provides a view of

the efficiency of the release plan in satisfying feature implementation requirements. From figure 4 it is clear that using inspections offers higher resource utilization and lower RUF values compared to the baseline scenario S1. Scenario S3 shows consistent improvement over baseline scenario S1 in resource utilization for all tasks. So with introducing new technology the resource usage shows visible improvement and has high impact on the quality of the release plan.

6.5   Summary and Limitations

In this paper, our approach does not focus on the financial or macro-organizational view of technology adoption. Rather, we concentrate on the strategic planning level of concerns in the process of technology adoption. It can serve as a new perspective for software industry in adopting new technologies. From a preliminary illustrative case study, we propose and analyze a data-driven planning prototype that takes technology change into consideration. We attempt to provide a paradigm for the technology change management process. An illustrative case has been studied using a sample project with 100 feature requirements, 91 stakeholders, 25 employee involvement. Four releases are planned with two planning criteria using the release planning framework EVOLVE II. In three different scenarios the results of introducing new technology Inspections is studied. To the best of our knowledge studying impact of new technology adoption on release planning for software engineering has not been highly focused in earlier literature. We focused to perform this study using specific instance of technology software inspection in a specific instant of release planning framework EVOLVE II.

This work has been conducted to analyze the impact of adopting a new technology in an existing release plan. The real life project integration was not available. Resource limitation and lack of real project integration is a threat to the validity. In this project the term new technology is narrowed down to engineering refinement category and further to software inspection (SI) technique for simplicity. All the studies and estimation models are created for SI. Number of factors that may affect a new technology adoption have considered out of our study scope. Nature of an inspection tool cannot represent the nature of all new technologies. To discuss about a new technology and its impact an empirical study have to be conducted which considers varieties of technology of different genre. Moreover for SI tool the rework elimination is considered equal for all type of tasks for simplicity. All the results are obtained in artificial experimental laboratory setup. So the concluding statements presented in this paper can help in future research by providing a direction but are subject to be verified by proper empirical study with real life projects integrations.

7   Conclusions and Future Research

The illustrative case study and the research work presented earlier results in three main conclusions. These conclusions might contribute in future research work, but are planned to be further validated by empirical studies with real life projects.

- Conclusion 1: Adopting new technology effects release planning decisions. Technological change impact can be considered as a potential parameter for release planning process that can substantially affect the release plan and might results in improvement.
- Conclusion 2: Adopting new technology effects release plan performance. It is a trade-off that may bring short-term negative effect but if it is well incorporated it normally brings benefits in the long run.
- Conclusion 3: If technology changes can be described by additional and revised effort estimates for feature implementation, the planning can be performed by the existing method EVOLVE II.

But in our paradigm in this paper, we treat technology as a new technique ignoring many aspects of technology and focusing on the impact of product requirements and management concern. We model the factor *initial technology acquisition cost* as training efforts assigned to each feature and later serving as an input for the planning process. Long-term effect on quality, time to market, or cost of the organization's products and services, are not included in our study.

Future research is intended to conduct on real life practical data through proper empirical studies. An attempt can be taken to justify and prove the statements presented above in respect of an empirical study done on different types of new technology adoption in real life projects. And if the study requires, a future researcher might refine the estimation models stated here to make it more appropriate to the real world. In addition, future research will enlarge the notion of *technological change* to make it more broadly applicable. Besides inspections, process changes or introduction of tools for testing and development could be introduced in a similar way.

Acknowledgment

References

1. Ruhe, G.: Product Release Planning: Methods, Tools and Applications. CRC Press (2010).
2. Muller, A., Merlyn, P., Valikangas, L.: Metrics for innovations: Guidelines for developing a customized suite of innovation metrics. Engineering Management Review, IEEE. 33, pp. 55.
3. Zelkowitz, M.V., Wallace, D.R., Binkley, D.W.: Experimental validation of new software technology. Series On Software Engineering And Knowledge Engineering. 12, pp. 229–263 (2003).
4. Goel, R.K., Rich, D.P.: On the adoption of new technologies. Applied Economics. 29, pp. 513-518 (2010).
5. Kang, K.C., Lee, J., Donohoe, P.: Feature-Oriented Product Line Engineering. Software. 19, pp. 58-65 (2002).

6. Greer, D., Ruhe, G.: Software release planning: an evolutionary and iterative approach. Information and Software Technology. 46, pp. 243--253 (2004).
7. Karlsson, J., Wohlin, C., Regnell, B.: An evaluation of methods for prioritizing software requirements. Information and Software Technology. 39, pp. 939–947 (1998).
8. Karlsson, J.: Software requirements prioritizing. Proceedings of the Second International Conference on Requirements Engineering. pp. 110–116 (1996).
9. Ruhe, G., Greer, D.: Quantitative Studies in Software Release Planning under Risk and Resource Constraints. Proceedings of the 2003 IEEE International Symposium on Empirical Software Engineering (ISESE 2003). pp. 262 – 271 (2003).
10. Ruhe, G., Ngo-The, A.: Hybrid Intelligence in Software Release Planning. Journal of Hybrid Intelligent Systems. 1, pp. 99-110 (2004).
11. Brown, A.W., Wallnau, K.C.: A framework for evaluating software technology. Software. 13, pp. 39-49 (1996).
12. Brown, S.A., Chervany, N.L., Reinicke, B.A.: What matters when introducing new information technology. Communications of the ACM. 50, pp. 91-96 (2007).
13. Huggett, M.: Does productivity growth fall after the adoption of new technology? Journal of Monetary Economics. 48, pp. 173-195.
14. Sauter, V.L.: Information Technology Adoption by Groups Across Time. International Journal of e-Collaboration. 4, pp. 51–76 (2008).
15. Brown, S.A., Massey, A.P., Montoya-weiss, M.M., Burkman, J.R.: Do I really have to? User acceptance of mandated technology. European Journal of Information Systems. 11, pp. 283-295.
16. Phaal, R., Farrukh, C., Mitchell, R., Probert, D.: Starting-up roadmapping fast. Engineering Management Review. 31, pp. 54-54 (2003).
17. Yoon, B., Phaal, R., David, P.: Morphology analysis for technology roadmapping: application of text mining. R&D Management. 38, pp. 51-68 (2008).
18. Macdonald, A.F., Miller, J., Brooks, A., Roper, M., Wood, M.: A Review of Tool Support for Software Inspection Scope for Tool Support. Seventh International Workshop on Computer-Aided Software Engineering. pp. 340 - 349 (1995).
19. DeBaud, J.M., Laitenberger, O.: An encompassing life cycle centric survey of software inspection. Journal of Systems and Software. 50, pp. 5–31.
20. Jones, C.: Software Engineering Best Practices. McGraw-Hill Pro-fessional Publishing, New York, NY, USA (2009).
21. Fagan, M.: Advances in Software Inspection. IEEE Transactions on Software Engineering. 12, (1986).
22. Hanakawa, N., Morisaki, S., Matsumoto, K.: A learning curve based simulation model for software development. 20th international conference on Software engineering. p. pp. 350--359 (1998).
23. Bartel, A.P.: Formal Employee Training Programs and Their Impact on Labor Productivity: Evidence from a Human Resources Survey. (1989).
24. ReleasePlanner, www.releaseplanner.com.
25. Boehm, B., Basili, V.R.: Software Defect Reduction Top 10 List. Computer. 34, pp. 135-137 (2001).
26. Anderson, P., Reps, T., Teitelbaum, T., Zarins, M.: Tool support for fine-grained soft-ware inspection. Software. 20, pp. 42-50.
27. Basili, V.R.: Evolving and packaging reading technologies. Journal of Systems and Software. 38, pp. 3-12 (1997).

Are my Features Innovative Enough?
– A Multi-Variable Innovation Strategy Model Proposal

Björn Regnell

Dept. of Computer Science, Lund University
`bjorn.regnell@cs.lth.se`

Abstract. A successful strategy for release planning is critical to an organization that offers software innovations to a market. This paper proposes the INNOREAP model, aimed at supporting the analysis of different feature selection strategies when trading off feature innovativeness against other innovation-related feature properties, such as effort, lead time, and revenue. Potential applications and limitations of the model are discussed in relation to directions of further research.

Keywords: release planning, software engineering, product management, innovation strategy, requirements engineering, new product development

1 Introduction

The question in the title regards whether the features of a release plan [1], [3] are representing an appropriate mix between incremental improvements demanded by the current market and novel product extensions that can increase future market shares. The answering of these types of questions is expected to be in the competence of a product manager [8] when carrying out strategic decisions-making for technology investments in future software products [5]. Finding answers is, however, inherently difficult and involves human expert judgment with significant uncertainty [6], [5]. Assessment of innovativeness before getting feedback from the market is particularly challenging [4].

This paper proposes a model that is intended to be both simple and useful in reasoning about different strategies for release planning, when balancing different innovation-related properties. The model explicitly includes innovativeness [2] (as one of multiple, inter-related decision variables), which to the best of our knowledge is not part of existing release planning models, cf. the systematic literature review on release planning by Svahnberg et al. [7]. The model is subsequently called the INNOREAP model denoting *Innovation strategy in Release Planning*. The purpose of this paper is to share initial ideas and to trigger discussions on if and how the model can be useful, as a starting point for further model development, and trials in conjunction with existing release planing models in particular and New Product Development models in general.

The paper is structured as follows. Section 2 defines INNOREAP. Section 3 discusses potential applications and limitations. Section 4 concludes the paper.

2 The INNOREAP Model

The basic idea of the INNOREAP model is to provide a framework of concepts for reasoning about innovation strategy in release planning. As the release planning problem

involves uncertain estimates of e.g. effort and business value, one could argue that a model should be rather coarse-grained; a very detailed model may not only be unnecessarily complex, but may also impose a false sense of accuracy.

INNOREAP reduces the release planning problem to a binary decision problem, where the product manager is faced with choosing only one out of two classifications of features for a limited set of feature properties. Thus decision-making is reduced from requiring ratio- or ordinal scale estimation to one binary choice per decision variable.

Decision variables. How many, and which decision variables to include in an innovation strategy model for software release planning is open to investigation, and the trade-off between expression power and simplicity is most likely context dependent. The initial version of INNOREAP proposed here includes four decision variables that may be of relevance for making trade-off decisions in innovation strategy management:

D : the *Degree of innovation* of a feature,
E : the *Effort* needed of human development resources in feature implementation,
A : the *Allocation span* of resources over releases that a feature require, and
R : the *Revenue* contribution of a feature.

For each of these decision variables only two different values are defined in order to reduce the general decision problem to a simpler 4-criteria binary choice problem. Assume that we have a feature universe $\mathcal{F}$ of N candidate features: $\mathcal{F} = \{f_k\}, k \in [1..N]$ and each feature f_k can have a property X. We express that it belongs to a set of all feature candidates with property X by the notation $\mathcal{F}_X$ where $\mathcal{F}_X \in \mathcal{F}$ and X is the first letter of the name of the property. We now make the following definition for our four feature properties D, E, A, R respectively:

Definition $D(f_k) \in \{Upgrading, Innovative\}$ A feature f_k belongs to the set of all upgrading features $\mathcal{F}_U$ if it gives an incremental improvement of existing product functionality and/or quality. In contrast, a feature is innovative and thus in $\mathcal{F}_I$ if it is judged to represent a novelty on the market that brings previously unseen but significantly valuable functionality and/or quality to current and/or future customers. Thus an innovative feature is a more radical extension of a software product compared to an upgrading feature.

Definition $E(f_k) \in \{Big, Small\}$ Let $e \in [0..1]$ represent a threshold ratio and $\mathbb{E}_{lim}$ the limiting available effort for the next release measured on a ratio scale of e.g. person hours. Let also $\mathbb{E}_{f_k}$ represent the estimated effort of feature f_k. A feature is defined to belong to the set of all big effort features, $f_k \in \mathcal{F}_B$, if it has an estimated effort above the threshold $e\mathbb{E}_{lim}$, thus $E(f_k) = Big$ if $\mathbb{E}_{f_k} > e\mathbb{E}_{lim}$, and vice versa: $E(f_k) = Small$ if $\mathbb{E}_{f_k} \leq e\mathbb{E}_{lim}$.
$\mathbb{E}_{tot} = \sum_{\mathcal{F}} \mathbb{E}_{f_k}$ is the total effort required by all candidate features $\mathcal{F}$.
In practice, $\mathbb{E}_{tot}$ almost always exceeds $\mathbb{E}_{lim}$, hence the need for release planning.

Definition $A(f_k) \in \{OneRelease, MultiRelease\}$ A feature f_k belongs to the set of all one-release features $\mathcal{F}_O$ if it is possible to fit the feature into the next release as it

is estimated not to allocate more than all available resources and its implementation is estimated to be feasibly completed before the upcoming next release date. A multi-release feature, on the contrary is not feasible to fit within the constraints of the next release. Such constraints can be e.g. its currently decided release date, the availability of required qualified resources, the capability of the current platform and software architecture etc.

Definition $R(f_k) \in \{HighRevenue, LowRevenue\}$ Let $r \in [0..1]$ represent a revenue decrease threshold and let $\mathbb{R}(F)$ denote the total revenue of a release if it would include all features in some set $F = \{f_k\}, k \in [1..n], F \subseteq \mathcal{F}$. We define a feature to be a high-revenue feature, if its exclusion render a relative revenue decrease normalized to the total revenue that is above the threshold r:

$$R(f_k) = HighRevenue \quad \Leftrightarrow \quad \frac{\mathbb{R}(\mathcal{F}) - \mathbb{R}(\mathcal{F} - f_k)}{\mathbb{R}(\mathcal{F})} > r$$

The above definitions entail eight subsets of $\mathcal{F}$, namely $\mathcal{F}_I, \mathcal{F}_U, \mathcal{F}_S, \mathcal{F}_B, \mathcal{F}_O, \mathcal{F}_M, \mathcal{F}_H$, and $\mathcal{F}_L$ respectively. The are pairwise complete partitions of $\mathcal{F}$ so that $|\mathcal{F}_B \bigcap \mathcal{F}_S| = 0, |\mathcal{F}_H \bigcap \mathcal{F}_L| = 0$, etc., and $|\mathcal{F}_B \bigcup \mathcal{F}_S| = N, |\mathcal{F}_H \bigcup \mathcal{F}_L| = N$, etc.

We also define $\mathcal{F}_{XYZW}$ to denote the subset that only has features for which all distinct properties X, Y, Z, and W holds. As there are four properties with exactly two values each, we have $2^4 = 16$ different such sets that build up the candidate feature universe $\mathcal{F} = \bigcup_x \mathcal{F}_x$, where $x \in \{ISOH, USOH, IBOH, UBOH, ... \text{ etc }\}$, using first letters of decision variable values for brevity.

Properties of sets of features. We define the following properties of an arbitrary set of features $F \subseteq \mathcal{F}$, where $n = |F|$ is the number of features in F, and $F_X \subseteq \mathcal{F}_X$. The first four feature set properties defined below are all in the $[0..1]$ interval:

Definition *innovativeness*$(F) = |F_I|/(|F_I| + |F_U|) = |F_I|/n$ The innovativeness of a set of features is defined as its ratio of innovative features.

Definition *effortfulness*$(F) = |F_B|/(|F_S| + |F_B|) = |F_B|/n$ The effortfulness of a feature set is defined as its ratio of big effort features.

Definition *preallocation*$(F) = |F_M|/(|F_M| + |F_O|) = |F_M|/n$ The preallocation of resources beyond the current release in a feature set is characterized by its ratio of multi-release features.

Definition *gainfulness*$(F) = |F_H|/(|F_H| + |F_L|) = |F_H|/n$ The gainfulness of a feature set is defined as its ratio of high-revenue features.

Definition The predicate *isOverscoped*$(F) \in \{\mathbf{true}, \mathbf{false}\}$ is true if the sum of all efforts of the features in F exceeds the limiting available effort of the next release, $\sum_{f_k \in F} \mathbb{E}_{f_k} > \mathbb{E}_{lim}$. To determine if this predicate is true, we need ratio scale estimations of efforts of all features in F.

Definition The predicate $isOverscopedBig(F) \in \{\mathbf{true}, \mathbf{false}\}$ is true if $|F_B| > \frac{1}{e}$, where $|F_B|$ is the number of big features in F and e is the effort ratio that is used to define the distinction between big and small features. This definition seems reasonable because if we would assume that all big effort features are as low as the threshold effort of eE_{lim} we need to ensure, in order not to overscope with big features only, that

$$\sum_1^{|F_B|} eE_{lim} < E_{lim} \quad \Leftrightarrow \quad |F_B| eE_{lim} < E_{lim} \quad \Leftrightarrow \quad |F_B| e < 1$$

To determine if this predicate is true, we only need to make binary estimates for the features in F to determine if they are big or small with respect to e, rather than the ratio scale estimates needed for the $isOverscoped$ predicate. Note however the limitation that the more easily determined $isOverscopedBig$ predicate does neither tell if the next release is overscoped by small features only, nor if it happens to be overscoped by a mix of small and big features, even if $isOverscopedBig$ is false.

Strategies in release planning. We subsequently define a release planning strategy as an ordering of decision variables that reflect a chosen relative importance of properties, with the aim to support exploration of alternative strategies (discussed in Section 3):

Definition of a release planning strategy. Let S_x be an M-tuple of $(x_m)_1^M$, $1 \leq M \leq 4$, representing an ordering of the feature properties as defined by one or more of our four binary decision variables x_m belonging to either one of $\{I, U\}$, $\{S, B\}$, $\{O, M\}$, or $\{H, L\}$. E.g. the 2-tuple $S_{(I,H)} = (Innovative, HighRevenue)$, or S_{IH} for short, denotes the strategy where high-revenue features are prioritized less than innovative features. If less than four feature properties are in x, then the ordering is said to be *partial* and not complete, representing that e.g. S_{IH} does not say anything about if the decision variable Allocation span is more important than Effort. If a strategy does not include a certain feature property, then that missing property is considered less important than all properties included in the strategy. Thus, the strategy S_{IH} prioritizes innovative features over e.g. small effort features.

Definition of strategic property inversion. For a strategy S_x we define an *inversion* of the m-th tuple element to entail a flip of the value of x_m to its other value in the binary domain of that element. For example, if we invert the 2nd tuple element of the 3-tuple strategy S_{UHO} we get the strategy S_{ULO}. We denote an inversion of a feature property using a bar over the property, e.g. $S_{U\bar{H}O} = S_{ULO}$

Definition of compliance between a strategy and release plan. We represent a candidate release plan simply by a list of candidate features $C = (f_1, f_2, ... f_n)$ chosen from the set of all features $f_i \in \mathcal{F}$, ordered in some priority order. For a given strategy S_x, we say that S_x and C are *compliant* with each other if the ordering of features in C is according to the (partial) ordering of the feature properties in x. This is denoted using a left-right arrow $S_x \leftrightarrow C$, entailing a predicate $\leftrightarrow$ that is

true if S_x and C are compliant. For example, assume that a candidate release plan C has $E(f_1) = Small$ and $E(f_2) = Big$. Our definition of compliance then entails that this particular C is not compliant with the strategy S_B, as big effort features comes after small ones, while strategy S_B prefers big features over small.

3 Discussion on Applications of INNOREAP

Is INNOREAP a good model? Good in what sense? This remains to be investigated further. In particular, it would be interesting to study the application of INNOREAP to the analysis of consequences of release planning strategies, as discussed subsequently.

Strategy consequence explorations can be made by deriving the feature property distribution after sorting a subset of all candidates so that it is compliant with some strategy and selecting the first n features as a candidate release. A feature distribution of a 2-property strategy can be visualized e.g. using a 4-quadrant, as proposed in Figure 1, showing the feature distribution of a feature set in relation to a certain strategy, where the number of features that has the properties of all combinations of property inversions. This show the number of features that are "best" according the strategy in the upper left corner, and the number of features that are "worst" in the lower right corner.

$|F_X \cap F_Y|$	$|F_X \cap F_{\bar{Y}}|$
$|F_{\bar{X}} \cap F_Y|$	$|F_{\bar{X}} \cap F_{\bar{Y}}|$

Fig. 1. An INNOREAP 4-quadrant of a feature set F for the strategy S_{XY}.

As an example, a product manager may be interested in assessing a certain set of features in relation to the following complementary strategies: the strategy S_B implying a concentration of effort to a few big features versus the strategy S_S for spreading available effort on a larger number of small features, combined with, e.g. either promoting a high revenue strategy S_H or promoting innovation S_I.

If we would construct different release plans that each comply with the aforementioned strategies respectively, the strategic 4-quadrant can be drawn for both S_{HB} and S_{IB}. This gives strategic 4-quadrants with feature distributions of the form:

HB	HS
LB	LS

and

IB	IS
UB	US

respectively. Also, a feature selection may be "optimized" for some of the previously defined feature set properties to get a top-n feature list in terms of e.g. maximal gainfulness, maximal innovativeness, minimal effortfulness or minimal preallocation. Would it not always be best to try to maximize gainfulness (and thus hopefully maximize profitability as well) by choosing the S_H strategy over S_I? Not necessarily, as a next release with a high innovativeness may pave the way for increased market shares that in the long-run will give higher profitability of future releases.

As feature properties may be inter-dependent, the distribution of features in different quadrants may be skewed. By depicting several 2-property strategies in different 4-quadrants, the balancing of different preferences can be discussed among stakeholders and used as input to product managers' decision-making, when trying to achieve a total, pragmatic balance of all aspects that are important for the next and future releases.

4 Conclusion

This paper proposes a model of release planning strategies that aims to give precise meaning to the concepts of degree of innovation, effort, resource allocation span, and revenue, that in turn can be assumed to be related to future market shares, cost, lead-time, and profitability respectively. These are important, inter-related factors when deciding what to invest in before the next release date and what to postpone to future releases, given limited resources and other constraints. The answer to the question in the title of this paper is more valuable if feature innovativeness is not only assessed in isolation, but in trade-offs with other important factors. INNOREAP can be applied to analyze various *combinations* of the aforementioned factors, in order to trade-off innovation in feature selection against e.g. concentration on some big features for a coherent and innovative release theme, minimizing cost or maximizing profitability.

The INNOREAP model is a crude simplification of real-world release planning, by reducing it to as set of binary choice problems. This simplification of course entails many limitations. For example, constraints among features are not taken into account. Other important feature properties may need inclusion, e.g. ability to keep existing customers or to attract new ones. Empirical studies are needed to evaluate INNOREAP in practice, and further studies are needed to understand which simplifications are over-simplifications, and which simplifications that a product manager appreciate when trying to sketch the big picture of the consequences of release planning decisions.

References

1. Brinkkemper, S., Ebert, C., Versendaal, J.: Proceedings of the first international workshop on software product management. In: Software Product Management, 2006. IWSPM '06. International Workshop on. pp. 1 –2 (sept 2006)
2. Crossan, M.M., Apaydin, M.: A multi-dimensional framework of organizational innovation: A systematic review of the literature. Journal of Management Studies 47(6), 1154 – 1191 (2010)
3. Ebert, C.: The impacts of software product management. Journal of Systems and Software 80(6), 850 – 861 (2007)
4. Regnell, B., Höst, M., Nilsson, F., Bengtsson, H.: A measurement framework for team level assessment of innovation capability in early requirements engineering. Product-Focused Software Process Improvement pp. 71–86 (2009)
5. Regnell, B., Brinkkemper, S.: Market-driven requirements engineering for software products. In: Aurum, A., Wohlin, C. (eds.) Engineering and Managing Software Requirements, pp. 287–308. Springer Berlin Heidelberg (2005)
6. Ruhe, G., Saliu, M.: The art and science of software release planning. Software, IEEE 22(6), 47 – 53 (nov-dec 2005)
7. Svahnberg, M., Gorschek, T., Feldt, R., Torkar, R., Saleem, S.B., Shafique, M.U.: A systematic review on strategic release planning models. Information and Software Technology 52(3), 237 – 248 (2010)
8. 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. 2006 International Workshop on Software Product Management (IWSPM'06) pp. 3–12 (2006)

Benchmarking Bundling Practices in the Software Industry

Joey van Angeren, Rick van Bommel, Catherine Arupia, and
Sjaak Brinkkemper

Department of Information and Computing Sciences, Utrecht University
Princetonplein 5, 3508 TB Utrecht, the Netherlands
`{j.vanangeren,r.a.a.vanbommel,c.arupia,s.brinkkemper}@uu.nl`

Abstract. Pricing comprises a crucial part of software product management. One strategy to follow is bundling; the sale of two or more products or services as one package. So far, little is known about the use and acceptance of bundling as a pricing tool within the software industry, especially when it comes to combining products and services into one package. In this paper we present the results of a small sample survey conducted with software companies, to both identify the bundling strategies that are employed and to let these companies benchmark their strategies with their competitors. In total, twenty-three companies took part in a web survey. Amongst others, results show that currently 71% of the companies make use of bundling and the average size of their packages is equal to five components. The configuration of such a package proceeds by assembling components around the product or service that is closest to the core competence of the organization.

Keywords: software product management, software business, product software, software related serivces, pricing strategy, bundling

1 Introduction

Software product management is becoming ever more important, with many businesses relying on virtual products (e.g. software products) for their revenue. Software product management is defined as: "the discipline that governs a software product over its whole life cycle, from its inception to customer delivery, in order to generate the biggest possible value to the business" [4]. The responsibilities within the discipline of SPM cover areas, such as requirements engineering and release planning. Because SPM involves generating the biggest possible value to the business, the responsibilities also evolve around price models and pricing strategies. A proper pricing strategy is of vital importance for a successful software company [12], and therefore needs more attention.

One of the topics covered within the domain of pricing is product bundling. Most definitions of bundling originate from either marketing or economics. From a marketing perspective, Gultinan [7] defines bundling as: "the marketing of two or more products and/or services in one package at a special price. Stremersch

& Tellis [18] define product bundling from a more generic perspective as: "the sale of two or more separate products as one package". While multiple definitions exist, the core of the definition remains the same, being: the practice of selling two or more products or services as one package. So far, attention for bundling mainly stemmed from researchers out of the domains of economics and marketing. The main focus of these researchers has been on different methods to maximize revenues and profits out of bundling by optimizing the product and service mix for a package. Also the domain of marketing regards bundling as a main strategy to attract customers. Examples of these studies are the work by Bakos & Brynjolfsson [1], or more recently by Gurler, Oztop & Sen [8].

Because of the intensive service flows within the product software industry the definition of bundling is not entirely satisfying. According to Cusumano [3], the revenue of a software vendor is comprised out of products (i.e. software and hardware), maintenance and services. A product software company can, for example, supply their customers with a software product that requires hardware. A software product in this sense is: "a packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market" [19]. Apart from that, this product may also require an array of services which are related to these products or even a mainframe. These services range from implementation services and training to customer-specific customizations. Furthermore, this system requires maintenance. All these artifacts and services can be an integral part of one package. On the contrary, a software vendor can also decide to offer the software product almost free of charge. However, it requires specific implementation services from the vendor in order to utilize the product. We regard both of these situations as instances of bundling and therefore we will refer to both of these forms of delivery as a package. A package in this sense can contain any configuration of software, maintenance, services and hardware. The artifacts that are part of such a package, we refer to as components.

As already acknowledged by Penttinen [17] back in 2004, the concept of bundling is not often studied within the research domains of information systems, product software and software business in general. When bundling is studied however, the emphasis is on bundling essentially separate products in one package. Lehmann & Buxmann [13] name the Adobe Creative Suite as an example of this, since it apart from photo editing also contains software to create PDF files. Due to this lack of attention, this paper directs attention to bundling of software, services and hardware. Through a small sample benchmark survey with software vendors, service providers and system integrators we map the influence bundling has on pricing strategies for components of a package. In order to achieve this, topics, such as the way in which a package is composed, how revenue can be attributed to components that comprise this package as well as bundle strategies and how these findings relate to different company types are addressed. The findings presented in this paper can be utilized by organizations within the software industry to benchmark their performance or to embrace bundling. Out of a researchers perspective, this paper is the first step towards

filling the void on the edge of the research domains of bundling and product software business and management.

The remainder of this paper continues with an explanation of the research question and all relevant sub questions in section two. The third section contains a process description of the data collection and analysis. Due to the exploratory nature of this research, results will be presented in the fourth section. We will elaborate on the gathered data set and we will conceptualize this data. An analysis and interpretation of these results is presented in section five. Section six contains a summary of the encountered validity threats as well as statements about possible generalization of the results presented. In section seven we draw the main conclusions and provide suggestions for future research.

2   Research Questions

The main research question answered in this paper is as follows: "*What is the influence of bundling on pricing mechanisms for components of a product package?*". In addition to the main research questions, two sub questions have been derived.

1. **How does the composition of a package relate to the revenue per component?** - A package can consist of any configuration of components. Because of the variety of different service flows, prominent within the software industry, this can lead to a diverse set of configurations. This sub question will be answered by studying both the composition of the package of each participant and the revenue each component represents within this package.
2. **What is the relation between the composition of a package and over- or underpricing individual components?** - As noted by Mulhern & Leone [15], packages can be made more attractive for potential customers by over- or underpricing certain components. Within the software industry, this dynamic is ever more present. By answering this sub question, we will identify the relation between the composition of a package and over- or underpricing. We will also identify patterns within this relation.

Both sub questions will be answered in section four and five. The answers to the sub questions form the basis to provide an answer to the main research question. Contextual characteristics, such as the company type, industry type and market type are used to provide a further classification in answering all the research questions.

3   Research Design

For performing this research, we have chosen to conduct a benchmark survey, since for now few research has been performed on the subject of bundling within the software industry. Because of this, a broad input in the form of a web survey

suits the research objectives better than a more in-depth method, such as a case study [2]. Furthermore, this way we enable software companies to benchmark their performance, by delivering a benchmark report that compares their results to the entire dataset. The procedures employed for this research are similar to the ones used by Jansen, Brinkkempter & Helms [11].
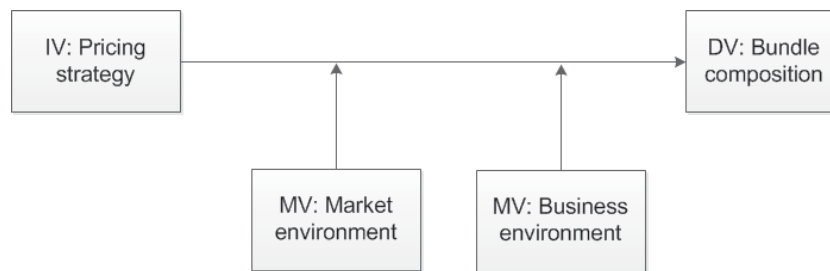
Fig. 1. Conceptual Model

As shown in figure 1, the core of the research involves investigating relations between the dependent variable bundling and the independent variable pricing strategy. However, various variables can have a moderating effect on this relation. These variables include the market and business environment. The variable market environment is measured by the perceived competitiveness of the market, market share and loyalty of the clients. The variable business environment is measured by the type of company and its dimensions. This research, for a large part, focused on quantitative data and for a smaller part on the amount of qualitative data that has been gathered. The qualitative part of this research aids in the effort of gaining new insights in rationale, while the quantitative part in this research intends to identify patterns and trends. Furthermore, basic quantitative methods were employed to prove or disprove the following hypotheses:

– **H1: The component being the core competence of the organization is the component that represents the highest annual revenue share.** Companies can focus on their core competences when doing business. Packages are then shaped around one main component, the component that the organization specializes in, the heart of the package and thus comprising the largest share.

– **H2: Software components as part of a package are purposely underpriced to stimulate sales of the total package.** The sale of software stimulates the sale of services within the package. Companies provide their software either at a lower price or for free to benefit the sale of services. The underlying assumption behind this is that it is easier for a potential customer to valuate the average market price for software and hardware. Doing this, the company attemps to benefit from recurring fees out of the intensive maintenance and support flows that are characterizing for the software industry [10].

3.1 Survey Design

The survey has been constructed out of three parts. Two parts of the survey refer to the measurement of factors related to the moderating variables; business environment and market environment. The other part aimed to elicitate all knowledge about bundling (i.e. the composition of a bundle) and pricing strategies (i.e. over- or underpricing). Each part consisted of around six to nine questions. The survey contained both open and closed questions and also included a number of multiple-choice questions and statements to be answered on a five point Likert scale [14].

Most multiple choice questions were employed to identify the decisions made with regard to pricing and package composition (e.g. the components that a bundle consists of or the revenue share that each of these components represents). Accompanying open questions provide insight into the rationale behind pricing strategies for components of a package. Most information related to the moderating variables is elicitated by employing categorized multiple choice questions as well as closed questions. At the end of the survey some statements on the basis of a Likert scale were included to measure customer loyalty and the attitude of companies with regard to bundling.

With regard to construct validity, findings from previous research [3,13,15] did form the basis for this survey. To further enhance construct and internal validity, the survey does adhere to the survey heuristics as defined by Fowler [5]. Furthermore, a pilot survey has been employed. Two participants have been asked to complete the survey to ensure the questions in the survey are unambiguous and result in the intended type of answers. These two participants have been excluded from further participation in this research. With the received feedback from these participants, approximately 5% of the survey was modified.

3.2 Data Collection and Sample Selection

To be able to conduct the survey, a digital survey environment has been set up with the open source software LimeSurvey, which includes Extensible Markup Language (XML) export functions in order to create a small bridge with other applications. The application that is connected with LimeSurvey is a self-written Hypertext Preprocessor (PHP) script that uses the open source library fPDF to generate the Portable Document Format (PDF) feedback report.

By means of a benchmark survey, potential participants found a strong incentive to participate in this research. A representative from a company that did complete the questionnaire and did submit his or her email address has received a simple benchmark PDF report. In this benchmark report, the scores of their company are compared with the scores of the total dataset.

Software vendors, system integrators and service providers have been contacted to take part in this survey. Desired respondents from a company were employees residing from management, sales or software product management functions. There were no restrictions in the geographic location of potential participants. Start-ups and companies employing less than five people were excluded

from the targeted participants. Another requirement was that their presence within the product software industry has to be significant and thus one of their main business functions.

Potential participants have been contacted through direct and indirect channels. Around twenty companies have been contacted directly through email and have been invited to fill in the questionnaire. Furthermore, the digital survey has been spread by means of professional network portals. Companies that are active within relevant groups on social networks have been invited to take part in this survey. The data gathering process took place for three weeks, starting in the middle of December until the end of the first week of January.

Because of the limited timeframe in which this research took place, the desired number of respondents is relatively low. To analyze results on a percentage scale we did at least need ten to fifteen respondents. In case of more than thirty responses, analysis could be more statistical and elaborative, however, to achieve a higher level of generalizability the number of respondents has to be higher than the indicated figures.

3.3   Data Analysis

For the purpose of data analysis both a qualitative and quantitative analysis have been fulfilled. For the qualitative analysis, rationales of respondents about over- or underpricing decisions and package composition were the data sources. This data, in term provices for a high-level analysis of the managerial decisions that have been made during the creation packages. Accordingly, these data are employed for conceptualization purposes, brief classification and are compared with findings described in previous research.

For the second part of the analysis, quantitative methods have been employed. This is done by a percentagewise comparison. This comparison analysis pinpoints variances that occur in groups between contextual information (e.g. core business, core industry) package configuration choices and resulting revenue. The latter is done through a pattern analysis. The contextual information encompasses two focus fields, namely the company and market information. Because of the relatively small sample size, we refrain from performing a more in-depth stastical analysis.

4   Results

In total, twenty-three product managers, CEOs, pricing managers or marketing and sales representatives did respond to the survey. Out of these twenty-three respondents, six were excluded from the final dataset. Their responses were either incomplete or they were not part of the target group. The final dataset consists of seventeen entries, 74% of the total number of respondents in the initial dataset.

4.1 Respondents

The majority of participants that completed the survey come from the Netherlands. A total of ten companies (59%) indicates their headquarters are located in the Netherlands. Three companies (17%) originate from the USA, two (12%) from the United Kingdom, one (6%) from Australia and one (6%) from France.

Because of targeting multiple company types we did ask participants to indicate the type of company that characterizes their organization the best. By far, the most companies are software vendors, followed by service providers. We did furthermore ask all participants to indicate what their core business is. The majority of respondents stem from the business process application segment, focusing on the development, integration or service provision of or for enterprise resource planning and business productivity applications. Figure 2 provides a full overview of how the respondents have been spread over the different categories.
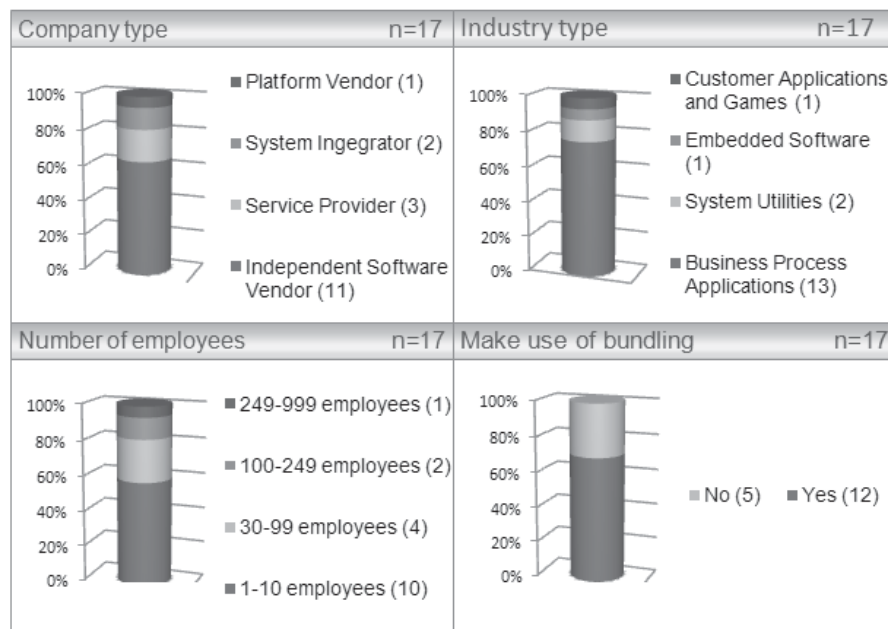

Fig. 2. Distribution of Respondents

Out of the seventeen respondents, twelve companies (71%) indicate to use bundling. The companies that do not employ bundling solely belong to the smallest company size. This corresponds with the maturity model for evolutionary growth of a software firm as proposed by Nambisan [16]. According to Nambisan, small companies are in a "start-up phase. Within this phase, the size of the product and service portfolio is limited, as well as the relative size of the

individual offerings, since they focus on their core asset. In the case of a software vendor, this focus lies on software development and testing. As a consequence, bundling becomes less relevant because the firm lacks resources to even compose a package. Even though the small companies comprise 59% of the total dataset, the majority of companies indicates to use bundling, we can therefore conclude that bundling is a popular approach within the software industry.

This conclusion is supported by the overall opinion the respondents have about bundling. Even though not all the respondents are currently using bundling, when asked only 14% of them indicates to not see much added-value in such a mechanism to stimulate sales of packages. Measured on a five-point Likert scale the average satisfaction equals a score of 3.57.

4.2 Package Composition

An important aspect of bundling is the actual composition of a package. We have asked the respondents to indicate which components together comprise their most prominent (e.g., most popular or best selling) package. The alternatives they could choose from have been derived from the notion of Cusumano [3] that the revenue of a software firm comes from products, maintenance and additional services. In correspondence, we did ask the respondents to indicate what percentage of total package revenue, each of these components represented. For this we employed the following categories; hardware, software products, customer specific customizations, implementation, maintenance, training and consultancy. Figure 3 provides an overview of the answers provided by the respondents.

As shown in figure 3, most respondents comprise their packages out of a wide array of products and services, providing for a strong diversification of offerings. Just two companies include respectively two or three components into their main package. Two other companies indicate that their bundle even consists of six or seven elements. On rounded average, the number of components in a package is equal to five.

Because of the large number of components within the package, not all customers will buy the entire bundle; rather they decide to take part of this composition. Hitt & Chen [9] refer to this as customized bundling. This variant of bundling appears to be most applicable for software companies, since it offers higher flexibility compared to pure bundling or mixed bundling and therefore facilitates the creation of large diverse packages.

With regard to this package composition, software components and accompanying implementation services are ever present within the described packages. This also goes for the service providers and system integrators, who, in correspondence with their business model, diversificate their total offering by reselling an existing software product, accompanied by implementation services and customer specific customizations. This is supported by the corresponding revenue shares, where only a small percentage originates from software. Furthermore, most companies indicate to benefit from recurring fees. Benefiting from intense service and maintenance flows they accompany their software products with specific implementation, maintenance and training.
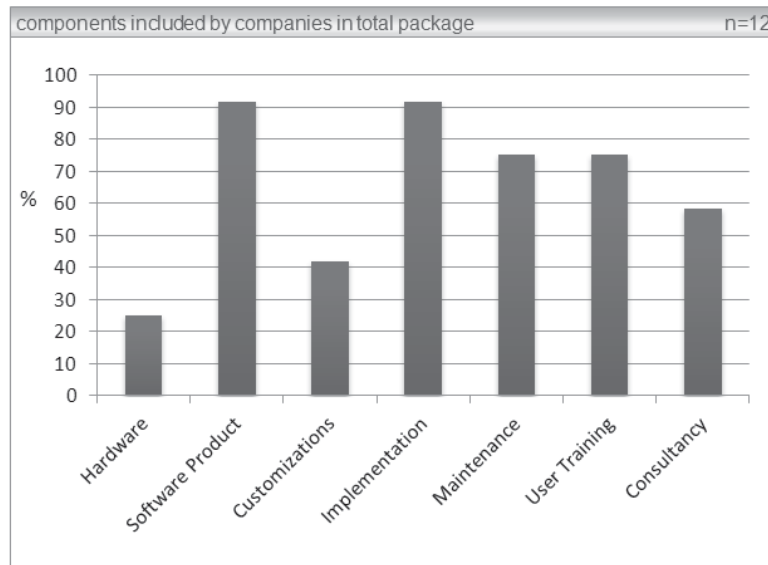
Fig. 3. Components included by software companies in total package

Physical goods, such as hardware and mainframes, are not included by the majority of the companies. Only two software vendors indicate to offer hardware components along with their software and services. This can either be caused by the large number of smaller companies that did fill out the questionnaire, or by the advent of Software as a Service. Also one system integrator indicates to resell hardware components.

4.3 Pricing Mechanisms

Companies were asked whether they under- or overprice (e.g. related to the market-price) software or service components. For software components, 17% indicates to overprice their software, 17% indicates to underprice while the rest conforms to a market-price. For services, only 8% indicates to underprice and also 8% indicates to overprice. Hardware components are also overpriced by 8% of the respondents. In total, five out of twelve companies indicate to use flexible pricing for individual components, the other seven companies indicate to not employ any flexible pricing mechanism. The firms that indicate to not employ flexible pricing mechanisms belong to the companies with a market share higher than the market share of companies employing these mechanisms. This supports the finding from Gallaugher & Wang [6] that there is a relationship between market share and pricing strategies. However, it can also be the case that these companies employ price bundling, a form of bundling in which the price of the total package is subject of fluctuation instead of varying prices of individual components.

The rationales for the use of flexible pricing strategies for different components vary per company. When underpricing, companies indicate to do this to either *increase market penetration* or to stimulate *tie-sales*. Companies that indicate to be active in highly competitive markets underprice part of their main components in order to compensate for this on the long-term through recurring fees, flowing in through long-term service contracts. Tie-sales are also employed to benefit from those recurring fees.

With regard to overpricing, it is found to be most applicable for the main component within the package or the core competence of the company. *Individual value propositions* and the *relative position of a product in relation to the products of competitors* are considered to be the main motives for overpricing.

5 Analysis

H1: The component being the core competence of the organization is the component that represents the highest annual revenue share.

Because of the relative small sample size we chose to apply and test this hypothesis with the largest company group within this dataset; the independent software vendors. In total, seven software vendors (e.g. the ones that did indicate to employ bundling) have been selected to test this hypothesis. To provide a detailed insight into the bundle composition decisions of software vendors, we created a table 1. For every component, this table describes how many software vendors did indicate to include this respective component in their offering. Furthermore, the respondents did provide an indication of the revenue contribution each of these componentns represents within the total package on an annual basis. To put these percentages into perspective, we computed the relative contribution to the total revenue for each of the components.

Table 1. Average distribution of revenue over package components as indicated by seven software vendors

Component	# times included in package	Contribution to revenue
Hardware	2	22,5%
Software product	7	41,50%
Customizations	2	2,5%
Implementation	7	19%
Maintenance	7	22%
Training	7	6%
Consultancy	4	9%

As can be noted from table 1, the majority of software vendors create packages that consist of five to seven different components. With a weighted average contribution of 41,50% to the total revenue, software comprises by far the largest

revenue share of the entire bundles offered. In the context of the hypothesis; software vendors surround their core competence, the software product, with an array of accompanying complementary services. This finding supports the formulated hypothesis.

The most prominent supporting components within the packages are implementation and maintenance. This corresponds with the revenue destribution for software firms as described by Cusumano [3]. Consultancy services and training comprise a very small part of the revenue per package. This supports the notion that these services are not very prominent within the package; they rather are employed for package completion or diversification. Another explanation could be that the majority of this task is being taken care of by partners. Doing so, the company can focus on its core competence, which lies in software development, assembly or implementation and maintenance. Furthermore, these additional services can be subject to underpricing, either to lower the costs of the total package or to make the package more attractive for customers.

The previously provided spreading of respondents in figure 2 reveals that the average number of employees for most companies is predominantly between one and ten employees. Accordingly, this indicates that there are not much resources left for dedicated support and implementation services. This either correlates with the employment of strategic partners to provide training and consultancy to benefit from an increased focus on software development or the "start-up" phase as elaborated on previously.

H2: Software components as part of a package are purposely underpriced to stimulate sales of the total package.

This hypothesis is tested by looking at the over- and underpricing decisions made by participants alongside with the rationale they did provide for this. As mentioned earlier, 20% indicates to purposely underprice their software product, while 20% indicates to overprice and the remainder does conform itself to market-price or has an internal view on pricing (e.g., they solely add a margin on top of their costs to determine the cost price). For services, only 10% indicates to underprice while also 10% indicates to overprice.

Based on these percentages and the small sample size, we cannot conclude whether there is a significant relation, if at all, between underpricing and the increased sale of accompanying components or the package as a whole. It only indicates that software companies are eager to employ bundling to benefit from flexible pricing mechanisms for individual components. This supports the findings from Mulhern & Leone [15] who noted that packages can be made more attractive by fluctuating prices of individual components. However, according to the responses, this decision is not related to the number of compositions that comprise the bundle as a whole.

The provided reasons for underpricing software, however, correspond to the line of thought behind this hypothesis. The very open and direct way in which the questions were answered accounted for some clear statements. In two particular cases underpricing was applied with a view on benefiting from recurring revenue. According to the respondents, these recurring revenues from mainte-

nance and service flows make up for revenue losses that are a consequence of underpricing, while these flows also provide for a higher revenue on a longer term. Furthermore, underpricing is also indicated to be attractive when aiming for market share. Underpricing is, especially in highly competitive markets or project bids, considered to be a good method.

These rationales are in correspondence with the hypothesis. This, however, is still not enough to prove the hypothesis. A large sample survey needs to be employed to enable in-depth statistical analysis. This statistical analysis needs to be employed to either prove or disprove this hypothesis, by drawing significant relationships between different variables.

6 Discussion

In this paper we presented the research findings based on data gathered through a small sample benchmark survey. To ensure construct validity, this survey has been designed in correspondence with findings from scientific literature. To enhance internal validity, we did commit the survey to the heuristics for an unambiguous survey design as defined by Fowler [5]. Furthermore, we did use two test companies to verify whether the questions were interpreted in the way they were meant to be. Participation in this survey proceeded anonymously to minimize the influence of personal biases of company representatives. Even though these measures have been taken, we cannot completely exclude that participants are biased, especially since pricing strategies are considered to be highly confidential.

We did invite software companies to take part in this survey through both direct and indirect channels. A relatively small number of companies has been invited by sending them an email, the majority however, did get access to the web survey by means of professional networking portal spreading. Because of spreading the survey by means of professional network portals, it is difficult to indicate the exact response rate. After removing the incomplete responses and responses from companies outside of the target group from the final dataset the total number of respondents is considered to be relatively small, even though the participants were offered an additional incentive for participating in the form of a benchmark report. In relation to this, the external validity, and therefore the generalizability of the results presented, is considered to be limited. Especially since, according to estimations by Statistics Netherlands, the number of product software vendors is estimated to be 2200.

7 Conclusion

In this paper we presented the results of a small sample survey, employed to investigate the applicability of bundling in the software industry. This because so far, few empirical studies have been conducted to address the bundling of products and services in the product software industry and because existing theories from marketing and economics are not satisfying for this industry. In

a relative short period of time, twenty-three companies took part in a web survey. Software vendors, system integrators and service providers were invited to participate in this research. In return for their participation they did receive a customized benchmark report.

Results show that 71% of the companies currently employ bundling as a pricing, delivery and marketing mechanism. Companies that do not employ bundling are small software vendors because their product and service portfolio is too small to implement bunding. The average package consists of five different components, meaning that diversification is considered to be important amongst the companies. This also supports the notion that customized bundling is the most applicable form of bundling for the software industry, since it aids in achieving a higher degree of flexibility with regard to package composition. In total, five out of twelve companies indicate to over- or underprice individual components within their package, meaning that bundling provides for more flexibility in pricing mechanisms. The respondents furthermore indicate to regard bundling as a valuable approach to product and service pricing.

A study of the composition of packages offered by software vendors shows that they construct a packages around their core competence; the software product. With an average contribution to the total package revenue of 41,50% the software comprises the largest part of the total package. A large number of additional services and maintenance, directly related to the product is then added to diversify the offering as a whole and to benefit from recurring fees. These recurring fees and market penetration play a vital role in over- or undepricing specific components. Two companies did indicate to underprice software to benefit from recurring fees out of maintenance and services on the long-term while three other companies indicate to underprice to increase the degree of market penetration. Overpricing occurs less often and is justified by individual value propositions.

The findings presented in this paper provide a first step towards expanding the body of empirical knowledge and data on bundling in the software industry. Since the number of respondents was limited, more research needs to be addressed to generalize the presented results to a larger scale. To enable researchers in doing so, the conceptual model that we presented in this paper can serve as a starting point. Since a coarse-grained conceptual model similar to the one we did present in this paper only suffices for smaller samples that do not involve in-depth statistical analysis, some adjustments have to be made. Larger samples also need to be addressed to perform in-depth statistical analysis in order to identify significant relationships between bundle composition and pricing mechanisms.

References

1. Bakos, Y., Brynjolfsson, E.: Bundling information goods: Pricing, profits and efficiency. Management Science 45(12), 1613–1630 (1999)
2. Blumberg, B., Cooper, D.R., Schindler, P.: Business Research Models. McGraw-Hill (2011)

3. Cusumano, M.: The changing labyrinth of software pricing. Communications of the ACM 50(7), 19–22 (2007)
4. Ebert, C.: Software product management. CROSSTALK The Journal of Defense Software Engineering Process Replication 22(1), 15–19 (2009)
5. Fowler, F.: Improving Survey Questions: Design and Evaluation. Sage Publications (1995)
6. Gallaugher, J., Wang, Y.: Understanding network effects in software markets: Evidence from web server pricing. MIS Quarterly 26(4), 303–327 (2002)
7. Guiltinan, J.: The price bundling of services: A normative framework. Journal of Marketing 51(4), 74–85 (1987)
8. Gurler, U., Oztop, S., Sen, A.: Optimal bundle formation and pricing of two products with limited stock. International Journal of Production Economics 118(2), 442–462 (2009)
9. Hitt, L., Chen, P.: Bundling with customer self-selection: a simple approach to bundling low marginal cost goods. Management Science 51(10), 1481–1493 (2005)
10. Jansen, S., Brinkkemper, S., Finkelstein, A.: Component assembly mechanisms and relationship intimacy in a software supply network. In: 15th International Annual EurOMA Conference, Special Interest Session on Software Supply Chains (2008)
11. Jansen, S., brinkkemper, S., Hemls, R.: Benchmarking the customer configuration updating practices of product software vendors. In: Proceedings of the Seventh international Conference on Composition-Based Software Systems. pp. 82–91 (2008)
12. Kittlaus, H., Clough, P.: Software Product Management and Pricing: Key Success Factors for Software Organizations. Springer-Verlag (2009)
13. Lehmann, S., Buxmann, P.: Pricing strategies of software vendors. Business and Information Systems Engineering 1(6), 452–462 (2010)
14. Likert, R.: A technique for the measurement of attitudes. Archives of Psychology 140, 44–53 (1932)
15. Mulhern, F., Leone, R.: Implicit price bundling of retail products: A multiproduct approach to maximizing story profitability. Journal of Marketing 55(10), 63–76 (1991)
16. Nambisan, S.: Software firm evolution and innovationorientation. Journal of Engineering and Technology Management 19(2), 141–165 (2002)
17. Penttinen, E.: Bundling of information goods - past, present and future. Sprouts: Working Papers on Information Systems 4(24), 1–26 (2004)
18. Stremersch, S., Tellis, G.J.: Strategic bundling of products and prices: A new synthesis for marketing. Journal of Marketing 66(1), 55–72 (2002)
19. Xu, L., Brinkkemper, S.: Concepts of product software. European Journal of Information Systems 16(5), 531–541 (2007)

Part II

REFSQ 2012 Empirical Track Proceedings

7 Preface

Editors

Joerg Doerr
Fraunhofer IESE, Germany, joerg.doerr@iese.fraunhofer.de

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

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

Empirical Track

Joerg Doerr[a], Norbert Seyff[b], Daniel Berry[c]

[a]Fraunhofer IESE, Germany; [b]University of Zurich, Switzerland;
[c]University of Waterloo, Canada

After the success of last year, we were given the opportunity to repeat the Empirical Track in 2012 and to expand it. Therefore, we 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 at the conference, and
- Empirical Research Fair Proposal: an empirical study that a researcher would like to conduct in an industrial setting or vice versa.

Overall we received fifteen high quality submissions, of which we selected eleven to be presented during the Empirical Track: one Alive Empirical Study, three Online Questionnaires and seven posters in the Empirical Fair.

1 Alive Empirical Study

The discussion at recent REFSQs have confirmed the strong need for empirical validation of the effectiveness for our Requirements Engineering (RE) methods, but the literature to date, including that of REFSQ, could show more of this validation. This lack is assumed to be at least partly due to the difficulty of finding and persuading the participation of a sufficient number of suitable experimental subjects. Therefore, REFSQ 2012 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 want to bring together the community of researchers and practitioners who are interested in empirical studies. Therefore, we selected the experiment titled *Do Stakeholders Understand Feature Descriptions?*, organised by Rumyana Proynova and Barbara Paech to be conducted at REFSQ 2012. This experiment aims to show how well a stakeholder can understand feature descriptions, and whether different forms of feature descriptions lead to different levels of understanding.

2 Online Questionnaires:

Online Questionnaires were a new kind of empirical study in the REFSQ Empirical Track, inspired by submissions we received in 2011. An online questionnaire (survey) is 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 2012:

- *A Survey on Empirical Requirements Engineering Research Practices* by Nelly Condori-Fernandez, Maya Daneva, and Roel Wieringa
- *A Survey on Requirements Engineering for Variability-intensive Software Systems* by Christian Manteuffel, Matthias Galster, and Paris Avgeriou
- *Requirements Engineering Techniques and Methods: An Online Questionnaire determining actual use in industry* by Richard Berntsson Svensson, Tony Gorschek:

The first two surveys received strong feedback by the REFSQ participants and the results are published in these proceedings. Because the last survey was aimed at a much smaller potential target group, only a few participants responded; so a description of the results in these proceedings was not possible.

3 Empirical Research Fair:

It is clearly understood in the RE community that case studies of industry projects are critical for our in-depth understanding of both: (a) the phenomena occurring in projects, processes, systems, and services and (b) the impact of our 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 seven proposals were presented on posters during the fair:

- *Tracing Requirements Interdependencies in Agile Teams* by Indira Nurdiani, Samuel Fricker, and Jürgen Börstler
- *What do you expect from Requirements Specifications? An Empirical Investigation of Information Needs* by Anne Gross
- *Applying Creativity Techniques to Requirements Elicitation: Defining an Enhanced EPMCreate* by Luisa Mich, Daniel Berry, and Victoria Sakhnini
- *Supporting Client-Developer Feedback Loops in Agile Requirements Engineering by means of a Mobile Requirements Engineering Tool* by Maya Daneva, Nelly Condori-Fernandez, and Norbert Seyff

- *Using E-mails and Phone Calls to Resolve Requirements Engineering Issues: Which Works Best and for Which Type of Issue?* by Maya Daneva
- *Patterns of Requirements-Related Communication* by Eric Knauss and Daniela Damian
- *Requirements Elicitation Driven by End-Users* by Alessia Knauss and Daniela Damian

Acknowledgements

Programm Committee

Ian Alexander	Scenario Plus, UK
Claudia P. Ayala	Technical University of Catalunya, Spain
Brian Berenbach	Siemens AG, USA
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
Natalia Juristo	Universidad Politécnica de Madrid, Spain
Søren Lauesen	IT-University of Copenhagen, Denmark
Nazim H. Madhavji	University of Western Ontario, Canada
Luisa Mich	University of Trento, Italy
Anne Persson	University of Skövde, Sweden
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

8 Alive Experiment

Alive Experiment Programme

Do Stakeholders Understand Feature Descriptions? A Live Experiment.

Rumyana Proynova and Barbara Paech

Software Engineering group, Institute for Computer Science, University of
Heidelberg, Germany
{proynova, paech}@informatik.uni-heidelberg.de

Abstract. [Context and motivation] Requirements engineers need feed-
back from stakeholders on planned system features. The simplest way
is to present feature descriptions to the stakeholders and ask for their
opinion. [Problem/question] The feedback is only valid if the stakehold-
ers' conception of the features represents the actual features reasonably
well. Due to the highly abstract nature of software, it is possible that
there is a mismatch between the stakeholders' idea of the feature descrip-
tion and the actual feature the software engineers intend to implement.
[Method/results] We conducted a live experiment during the RefsQ 2012
conference. We used a questionnaire to measure the mismatch between
the participants' understanding and liking of a list of software features
and the implementation of these features, shown as a screencast of a
system prototype. We found a correlation between the degree of un-
derstanding of a feature and the liking of that feature. There were no
significant differences between features presented in different formats.
[Contribution] This experiment shows first insights into the factors which
contribute to a stakeholder's understanding of a feature description, and
to his/her satisfaction with a software which contains these features.

1 Introduction

It is not feasible to involve end users in requirements elicitation in all projects,
even though this could lead to higher quality requirements. Factors like the un-
availability of end users (for example in global software development projects or
off-the-shelf software products with no designated end users) or limited budget
and resources, as well as company culture, can dictate that the requirements for
the software are derived from other sources. In order to ensure that these require-
ments are aligned with the needs of the end users, the requirements engineers
can let the users validate the requirements.

The constraints which preclude resource-intensive elicitation techniques are
likely to also preclude similarly resource-intensive validation techniques. A tech-
nique, which produces adequate results but requires a comparably low level of
effort, can enable early validation in projects where currently end users are not
involved until the very late stages of the project such as testing or even roll-out
of a completed product. Our research focusses on defining such a technique. The

experiment described in this article is part of the research needed for the creation of this technique.

The technique is based on the well-known marketing concept of using a questionnaire to measure customers' expected satisfaction with product features [11]. Such a questionnaire lists a number of features and the prospective customer indicates his or her expected satisfaction on a Likert scale. While marketing literature provides research on using this technique, it cannot be used for software products without substantial extensions. The abstract nature of software and the often intricate dependencies between features make it difficult for the questionnaire respondent to give meaningful ratings for the individual features.

While the questionnaire technique has some distinct advantages, such as making efficient use of the time of the requirements professional (as there are no separate interviews with each user) and delivering quantitative data, it also has its drawbacks. The communication between end users and requirements engineers is very limited, and there is no convenient channel for inquiries, clarifications and discussion. All information a user gets about a feature is its description. If users misunderstand a feature description, they may approve a feature they do not need, or declare that they need a feature which is in fact useless. They may also have ideas about how to improve the features, but may not inform the requirements engineers, because they are unsure about the proper communication channel.

For the communication between the requirements engineer and the end user to function properly, it is important that the requirements engineer asks the right questions, the users understand them in the way they were intended, and answer truthfully. This article describes an experiment which we conducted in order to gain more insight into the first two points. Our experiment does not address the possibility of users intentionally giving wrong answers.

In the next section, we describe research related to our experiment. In section 3, we describe how we designed our questionnaire and how we conducted the experiment itself. Section 4 describes our hypotheses, our results, and some explorative conclusions we made in addition to the predefined hypotheses. The last section presents our intended future work on this topic.

2 Related Work

Our experiment uses a questionnaire for measuring the users' satisfaction with a product. There are several techniques in marketing for similar types of analysis. Prominent ones are concept testing [10], conjoint analysis [7], importance performance analysis [9] and SERVQUAL [1].

None of these methods can be used directly with software features. Concept testing presumes that a customer is able to make a buying decision based on an advertisement presented on a magazine page. This is useful for products like toothpaste, which only have a few simple features, but is not feasible for software products. Conjoint analysis requires a customer to make pairwise comparisons between each possible level of many factors, for example a product can

have the factors *product warranty (levels: 1 year, 2 years)* and *support options (levels: telephone hotline, online ticket submission)* and the customer has to rank all four possible combinations. As the number of comparisons to be made grows exponentially with the number of factors present, it is not suitable for a products with multiple features. Importance-performance analysis needs easily quantifiable features, for example *fizziness* for a soda. SERVQUAL includes a hierarchical set of service qualities, which can not be mapped to a software product without changes. All four of these techniques assume that the name of a feature conveys enough information about it to be perfectly understandable for a customer, and the studies which use these methods are done on products with simple, well-known features where understandability is not a problem.

Our research is based on software features. Software features are an important part of how users understand software. They are often used in software specifications, especially in the context of software product lines. [2], [12]. Other types of specification can often be broken down into individual features, the way we do this in the experiment described here - we produce our features from a software specification written as user tasks [8] and user stories [3].

3 Experiment Design

The experiment was conducted during the empirical track of the RefsQ 2012 conference, with 56 conference visitors participating. Each participant received a questionnaire and was asked to answer the first part of it. This part contained software feature descriptions and questions about them. Then the participants were shown a screencast of a software application implementing the features from the first part, and had to answer questions about the feature implementation.

For the experiment, we created a software requirements specification and a software prototype implementing the specification. We chose to implement a personal finance software product for managing receipts. This choice had several advantages. First, this application could be made simple enough to cover the complete specification of functional requirements in the limited time available for the experiment. Second, this class of software product is not especially common, and we expected most of the participants to not have made experience with similar software products before. This reduced the chance of preconceptions formed through contact with similar software products to skew the results. Third, using this software does not require any special knowledge. Had we chosen a software supporting a process which is only performed in certain professional fields or hobbies, we would have had to control for the participants' experience in these fields or hobbies.

We used the features from our requirements specification for the features description in the questionnaire. We recorded a screencast of the first author using the software, and used this recording for the experiment. The questions in the questionnaire were derived from our research goals.

3.1 Research Goals

The purpose of this experiment was explorative research. We wanted to get first insights into how users understand feature descriptions and how a requirements engineer can create a questionnaire which reflects the true future satisfaction of an end user with given software features. Such an instrument can never be perfectly accurate, but we want to achieve a degree of accuracy high enough for it to be used in decision-making about which features should be implemented in a software product.

We had three main research goals. We wanted *to a*) understand to what degree an end user's satisfaction with an implemented feature correlates with a set of factors we assumed are connected to satisfaction, most importantly to what degree self-predicted satisfaction correlates with actual satisfaction; *to b*) to find further factors, beside the ones we assumed in the previous goal, which could possibly play a role in self-reported expected satisfaction; and *to c*) to test whether different feature presentation formats influence the understanding of features.

Correlation of end user satisfaction with proposed factors While marketing theory often works with complicated concepts to describe satisfaction, a questionnaire study has to use simple, universally understood concepts in a questionnaire, else it risks getting incomparable results due to a confusion of what is being asked. For this experiment, we chose one of the simplest concepts possible: liking. As we are interested not only in knowing self-reported liking, but also in how accurate the answers are, we also included questions about the understanding of what a feature is about. A misunderstood feature will lead to answers based on a false conception of the feature and therefore reduce the usefulness of a questionnaire. We asked about the liking and understanding both before and after the participants saw the feature implemented in a software demonstration. We formulated multiple hypotheses about the possible relation between how well participants believed they have understood a feature before and after implementation, and how much they liked it before and after implementation. Below is a list of these hypotheses, with a possible explanation for each hypothesis given in parentheses.

Hypothesis 1 Accurate understanding of an implemented feature is related to a feeling of understanding it before seeing it implemented. (Users know whether they have understood a feature).

Hypothesis 2 Liking of a feature before seeing it implemented is related to a feeling of understanding it before seeing it implemented. (Users like conceptions they understand).

Hypothesis 3 Liking of a feature after seeing it implemented is related to a feeling of understanding it before seeing it implemented. (Users like features which were clear from the beginning).

Hypothesis 4 The deviation in liking a feature before and after seeing it implemented is negatively related to a feeling of understanding it before seeing

it implemented. (The better users have understood a feature, the better they can predict whether they will like it. Alternatively, the causation could be reversed: The more the users like a feature, the better they are able to understand it with a short description).

Finding further relevant factors Unlike the other research goals, this one was based on qualitative research. We collected feedback about what participants thought influenced their answers to the questionnaire. There were no preformulated hypotheses for this goal.

Different feature presentations Software requirements can be documented and represented in multiple ways. Some of these are not suitable for use in a questionnaire for non-technical users. We considered several formats which were more or less self-explanatory. As we were concerned with a clear understanding by a user who does not have the possibility to ask for clarification, we decided to compare formats which are more or less close to a user's point of view. We included user stories and user tasks in our questionnaire.

We based our user stories on the guidelines given by [3]. They represent the type of requirements documentation used in modern agile software development approaches such as Scrum. Each user story is a short description of what the user can do with the system, which is supposed to be documented on a separate card and used primarily for release planning and developer task specification. A user story is always written from the point of view of the end user, it is closed and supports a user achieving a goal. The granularity is determined by the development process: user stories do not contain details, so each of them must be small enough to be sufficiently described in one or two sentences, and to be implemented by a developer within a single sprint. Each user story should be independent of other user stories. For the survey, we use each user story as a separate feature. While a user story card can contain constraints (non-functional requirements) or acceptance tests, we do not include these in the survey, as they do not have a direct analogue in the other two approaches.

For user tasks, we use the guidelines proposed by [8], specifically the so-called "task and support" approach. In this approach, the subtasks are combined with proposed solutions which can be implemented in the system. User tasks are written close to a user's point of view. Each task is based on a goal the user wants to achieve. Subtasks do not focus on either user or system; they "describe what the user and the computer do together" on a domain level. They are written in the imperative mood, to hide who does what. The solutions may specify what the system does, but this is not always the case. User tasks do not provide an ordered sequence of steps the way use cases or scenarios do. However, they package small substeps into bigger units, the tasks. Each task has one user goal and should have closure. This means that a task provides more context information to the user than formats which record each subtask-sized requirement independently. For the purposes of our survey, we define each solution from the Task & Support approach as a product feature.

We formulated several hypotheses on the differences which formats can cause in the answers.

Hypothesis 5 The closer a format is to the user's point of view, the better the feeling of understanding it before seeing it implemented. (Users feel they understand feature descriptions better when they are described from their point of view).

Hypothesis 6 The closer a format to the user's point of view, the better the understanding after seeing it implemented. (Users understand features better when they are described from their point of view).

Hypothesis 7 The closer a format is to the user's point of view, the less deviation is there in liking the feature before and after seeing it implemented. (Users can better predict their liking of a feature when it is described from their point of view).

3.2 Features

The requirements specification used for the experiment contained fifteen features. The complete specification in the user task format is shown in section 7. We tried to make the specification as realistic as possible. We did not polish it to be the best specification we can produce, but included features which we assumed were not very good from the end users' point of view. This was done in order to have variability in the features' liking, as we feared that, if all users like all features uniformly, we would not be able to see any effects. We also included a "double" feature, where the users had to rate two either-or alternatives, none of which were very user-friendly. The reason was that we wanted to force the participants to dislike at least one feature, again to have more variability. Also, the fact that one feature excluded the other was unusual (there was no obvious technical reason why both could not be implemented at the same time), and we assumed that some participants would be misled to believe that both features will be included in the software allowing the user to choose between them at runtime. We hoped that this would create a situation where users believe that they have understood the feature before seeing the implementation, but recognize that their understanding was not accurate after seeing the screencast.

Beside variability in liking, we also wanted to create variability in understanding. For this, we knowingly kept some features ambiguous, omitting important details. We feel that this made the situation more realistic. First, the literature on user stories and user tasks suggests that these descriptions should not contain very detailed information in order to prevent overspecification [8], [3]. Second, in industry projects we have been involved with, we have often seen features described at the same or at a more abstract detail level as the one we used, but never at a lower, information-richer detail level. Thus, including intentionally ambiguous feature descriptions made the experiment situation more realistic.

We developed a GUI protoype based on our specification and created a realistic test data set including actual receipts. The prototype was created in Java and had only rudimentary functionality, but, when used with the test data set,

it could create a convincing illusion of actually having the functionality described in the specification. We created screencasts of one experimenter using the features described in the specification, accompanied by an audio commentary explaining what is happening on the screen. These screencasts were shown to the participants during the experiment, as explained in the next section.

3.3 Questionnaire

The questionnaire consisted of four parts. The participants were given time to answer parts one and two, then asked to wait for a software demonstration.

Part one contained general demographic questions. Participants were asked about their expertise in requirements engineering and their experience with different requirements representation formats. The data from these answers is not evaluated for any of the hypotheses listed in the research goals; rather, it is used to describe relevant demographic factors of the experiment participants.

In part two, the participants were asked to read a requirements specification and answer questions about the features it contained. There were two types of questionnaires, which differed in this part. One type had user stories, each printed in a simple box without a number, representing a story card. The second type had two user tasks formatted as described in [8] with each solution containing a single feature. The wording of all features was equivalent in both formats except for rules prescribed by the format.

For each feature, participants were asked to answer two questions:

Question 1 "My conception of the way this feature will be implemented is ...", followed by a five-point Likert scale labelled clear/vague/non-existent (All Likert scales used in this questionnaire were five-point with only the first, third and fifth point labelled). This question was used to measure the feeling of understanding before seeing the implemented feature.

Question 2 "I think I will ... this feature when it is implemented", followed by a Likert scale labelled like/be indifferent/dislike. This question was used to measure liking before seeing the implementation.

These two questions were followed by a prioritization task. We have not yet completed the evaluations based on the data from the prioritization, so we do not report on it further in this article.

In part three, the participants again were shown each feature, with three new questions per feature. They were instructed to wait for a video demonstration of the features before answering the questions. Each demonstration contained two or three features, so that features which were solutions to the same task as defined in the user tasks were bundled in the same demonstration. Participants were given time to answer the questions about all features in a demonstration before the next demonstration was presented.

Question 3 "The feature corresponds ... to my previous conception", with a Likert scale labelled very well/somewhat/not at all. This question was used to measure accurate understanding.

Question 4 "The implemented feature differs from my previous conception in the following ways:", followed by empty space for a freely formulated answer. This question was used for generating ideas for other relevant factors (second research goal).

Question 5 "I ... the feature the way it is implemented now.", followed by a Likert scale labelled like/be indifferent/dislike. This question was used to measure liking of a feature after seeing the implementation.

Part four included one open-ended question, and a feedback section. In the question, we asked the participants to give us suggestions on how they would have improved the feature descriptions to improve their understandability. In the feedback section, we asked for freely-formulated feedback on any part of the experiment, the conduction, and the questionnaire. The answers from this part were used to generate new ideas for relevant factors (second research goal), as well as for recognizing possible threats to validity, and as information on how to structure a follow-up experiment.

4 Results

We used the statistical language R to evaluate the hypotheses stated in section 3.1. We could not confirm the hypotheses about the differences in the description formats, but found a good correlation in most hypotheses about the relations between the understanding and liking of features before and after seeing their implementation. The qualitative analysis of the open-ended questions as well as the feedback the participants gave during a discussion session at the conference led to interesting new ideas which we will explore in further experiments.

4.1 Correlation of End User Satisfaction with Proposed Factors

To this research goal, we calculated a correlation coefficient for hypotheses 1 through 4. As this is not a significance test for a parameter, we can not offer a significance level; rather, we can say that the higher the absolute value of the correlation coefficient, the more connection there is between the two phenomena. The results are listed in table 1. Figure 1 gives an overview of the correlations we found.
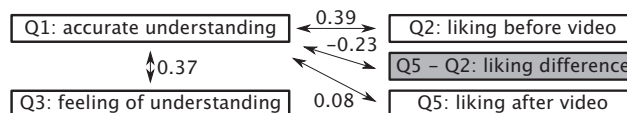
Fig. 1. Concepts measured and their correlations

We found that most hypotheses of this research goal, with the exception of hypothesis 3, were confirmed. The correlation coefficients were clearly different from zero (so the two factors are related), and the sign direction was as the

Hypothesis	Questions	Correlation
Accurate understanding of an implemented feature is related to a feeling of understanding it before seeing it implemented.	Q 1 and Q3	0.37
Liking of a feature before seeing it implemented is related to a feeling of understanding it before seeing it implemented.	Q 2 and Q 1	0.39
Liking of a feature after seeing it implemented is related to a feeling of understanding it before seeing it implemented.	Q 1 and Q5	0.08
The deviation in liking a feature before and after seeing it implemented is negatively related to a feeling of understanding it before seeing it implemented.	(Q 5 - Q 2) and Q1	-0.23

Table 1. Hypotheses about the connections between the concepts

hypothesis predicted. We conclude that a clear conception is an important factor in the participants' answers to a questionnaire with software feature descriptions. We propose the tentative interpretation that people tend to be unsure about what answer to give for a feature they can not picture clearly. We cannot claim that the data confirms this interpretation; we need more research, not based on pure correlation, to find out whether this interpretation is correct.

4.2   Finding Further Relevant Factors

We are interested to know what influences the answers the participants give to questions such as "Do you like this feature" and also their ability to envision how a feature will be implemented. The discussion and the answers to the open questions provided us with ideas about such factors.

Participant has encountered feature before. We chose an unknown software product, in order to avoid the situation that participants have preconceived notions about features based on their previous experience with such software. The discussion made us realize that this effect can emerge not only on the level of a software application, but also on the level of a single feature. The participants reported that they had a good understanding of features which they had previously encountered in unrelated software products, and that they had difficulty creating a clear concept of features specific to our software. The data confirms this claim: the features with the highest arithmetic mean in the answer to question 1 were the ones related to printing (feature 14, $\mu = 4.75$)[1], exporting a report (feature 15, $\mu = 4.44$) and saving data in a file (feature 7b, $\mu = 4.16$).

---

[1] As the results are measured on a five point Likert scale, the minimum possible score is 1, the maximum possible score is 5

This functionality is common in many types of software, and probably most of our participants have encountered it before.

Participant cannot imagine a good use for the feature. This was a comment left by one participant in the feedback part. He or she claimed that it was very hard to build a clear conception of the feature when he or she did not know why it could be useful or what they would want to use it for. This is an interesting comment which is in line with the recognized importance of rationale in requirements engineering [4]. In a follow-up experiment, we will include questions designed to measure the influence of this factor on the understanding.

Participant has no emotional attachment to the software product. This was suggested by a participant who claimed that he or she marked the "indifferent" option for all features not because of the relative merit of the feature as compared to other features in the description or to other possible features implementing a similar functionality, but because he or she found the software very "boring" and was indeed indifferent to anything which had to do with the software product we presented. This finding is in line with results of marketing research. First, modern theories of consumer satisfaction agree that satisfaction has both an emotional and a rational dimension. We chose to ask a single (expected) satisfaction question in this experiment, and formulated it using a word associated with emotions: *like*. It is possible that another formulation, such as one using a word associated with rational utility like *useful*, would have produced a different result at least from this user. However, marketing literature emphasizes that a customer should create an emotional connection to a product for satisfaction (as made popular by the AIDA principle - a customer's relation to a product starts with simple Attention, but it has to include Interest and Desire before they Act on it [5]), so maybe, while a rational formulation of the question could have produced different results, they would not necessarily have reflected the user's satisfaction well enough. This is an interesting connection we plan to research in more depth in future experiments.

Suggested improvements to feature presentation At the end of the questionnaire, we asked the participants to leave their assumed role of end users and reflect from the viewpoint of requirements specialists. We asked them to give us suggestions on how to improve the feature descriptions in order to achieve a better understanding and more accurate answers about the expected satisfaction. We received many suggestions, some of them given by more than one participant. We agree that all these suggestions have the potential to increase the understanding, but our background is a specific situation: a project with limited resources in an early phase of requirements elicitation and validation. Thus, we evaluated the suggestions on two criteria: the potential for increasing the users' understanding and the feasibility of a project team creating the suggested artefacts in this project phase. We will do further research on the suggestions which scored high in both dimensions.

Participants' suggestions for individual features While evaluating the answers to the open-ended questions, we observed that the participants often offered suggestions for improving a feature, or at least could articulate what they dislike in a feature. Without receiving any specific guidance, they often made remarks which could be clearly assigned to the desire for a different user interface or different functionality, and they made clear what they would like to see instead of our solution. This effect was especially strong in features which were intentionally designed to be user-unfriendly, but it also existed in other features. The users even noticed potential problems which we had not noticed in our specification, but which sounded like true problems (as compared to the desire to have something in a different way without being objectively better) and could be solved without much additional development effort (in the case that the application was actually implemented).

While we are aware that our participants were experienced software engineers and thus can be expected to come up with such suggestions at a higher rate than the general population, the suggestions they had were often simple, interaction-specific changes which are not beyond the reach of users without a background in software engineering. For example, a frequent suggestion was that the report should be exported in common office formats, such as Word of PDF. Other suggestions showed that the participants were evaluating the features from the point of view of a user and not of a software engineer, for example the frequent suggestion that the text recognition should function reliably enough that the user does not have to confirm and correct the results. While this is a very understandable desire of a user, we expect software engineers to consider technology limitations before writing such a suggestion. We found no indication that such limitations had been considered, except for one participant who did not suggest perfect recognition, but instead suggested that the software displays reliability numbers (how confident it is that a given recognized text string is correct). Thus, we expect users without experience in software engineering to be able to give similar feedback, even though we expect them to have a lower suggestion rate and a lower ratio of useful to infeasible feature change suggestions. This expectation has yet to be confirmed in future research before it is incorporated in our technique.

4.3 Different Feature Presentations

One of our research goals was to compare the liking and understanding between features described in different formats. The used formats were rather similar in their degree of closeness to the users' point of view, but differed in the amount of context offered, with user tasks bundling related features together and containing additional information per task, while the user stories were represented on unconnected "cards". Thus we expected to see differences between the two.

We did two comparisons in parallel. The first one was a straightforward t-test of the arithmetic mean calculated separately for both types of feature description. The second one was an ANOVA test, which, with only one pair of treatments, resulted in a simple t-test for the variance of both distributions [6]. In

the following, we call the participants who answered the user task questionnaire *user task group* and the participants who answered the user story questionnaire *user story group*.

Table 2 shows the exact formulation of our hypotheses. The column *Question* contains the number of the question which was used for evaluating the hypothesis. We evaluated by comparing the answers to this question given by the two groups. The last hypothesis included building a difference of the answers of two questions and comparing the mean and the variance of the difference. None of the hypotheses could be confirmed at a significance level of 5 %, neither when comparing the means nor when comparing the variances.

Hypothesis	Question	Result
The closer a format is to the user's point of view, the better the feeling of understanding it before seeing it implemented.	Q 1	not confirmed
The closer a format to the user's point of view, the better the understanding after seeing it implemented.	Q 3	not confirmed
The closer a format is to the user's point of view, the less deviation is there in liking the feature before and after seeing it implemented.	Q 2 - Q5	not confirmed

Table 2. Hypotheses about different feature presentations

We could not confirm any of the hypotheses we created for this research goal. Our conclusion is that the format in which feature descriptions are represented does not matter for the understanding or the liking of features, at least when we compare user tasks to user stories. The richer context of user tasks does not seem to lead to a better understanding. In the future, we plan to research the same hypotheses, but to include a format which is far removed from the users' point of view, for example sentence templates. The templates described e.g. in [13] begin with "The system shall" and continue to describe what the system does, instead of what the user does using the system.

4.4 Threats to validity

Conclusion validity We identified two possible threats to conclusion validity. First, we are not aware of any objective measures for the variables we measured, so we had to rely on self-reporting. Second, we ran a large number of hypotheses on the same data set. This is problematic for confirming a theory, as it lowers the actual significance level, but we used significance tests only for exploratory research and do not claim that the results are conclusive.

Internal validity The quality of the instruments we used for our experiment can also have compromised our experiment validity. First, users may have misunderstood questions due to ambiguity. We tried to counter this by presenting the questionnaires to coworkers not involved in the project and asking them

whether they understood the questions the way we intended them. Second, we used screencasts of the software. The users' true judgement of features may have been incomplete, because they did not work with the software themselves.

Construct validity The questions we asked may not be best suited to measure the concepts of understanding and liking. For example, some users may have found it hard to answer whether their understanding had been accurate, as they might have had difficulty remembering how they imagined the features before the screencast.

External validity The most serious problem with our experiment is that the participants were software engineering experts and not typical users with non-technical background. The situation we used in the experiment was also not perfectly realistic. The users did not know why they should use the software. The feature descriptions may also not represent user stories well enough. We created them as tasks first and "translated" them into user stories. It is possible that requirements written after user story principles would have had a different content or granularity.

5 Conclusions and Future Work

Our experiment is based on the assumption of a certain project situation - the need to validate features with end users who were not involved in the feature elicitation, under limited resources. We propose that in this case, a questionnaire can be used for the validation, and the experiment aimed at finding out how accurate such a questionnaire can predict the users' future satisfaction with the software. We had three main research questions. Two of them were based on assumptions we made, and tried to confirm a number of hypotheses related to each assumption.

We could confirm almost all hypotheses related to the assumption that the ability to build a clear conception of a future implementation based on just a feature description has a strong influence on the answers to an expected satisfaction questionnaire. We plan to continue research focussed on this understanding and ways to improve it, in order to reduce the undesirable influence of low understanding on the rating of features on a scale for expected satisfaction.

We could not confirm the hypotheses related to the assumption that the feature description format has an influence on the answers to an expected satisfaction questionnaire. This finding has important consequences for our proposed technique: if every format is understood well enough, the technique can be used in projects employing any format without the need to translate the requirements to a specific format suited to the technique.

The open-ended questions as well as the feedback section allowed us to find other possible factors influencing the accuracy of self-reported satisfaction expectation. We intend to measure the influence of these factors in future research.

The future research questions mentioned above will be addressed in a future experiment, conducted with university students without a software engineering background. The design of the new experiment will be altered in order to measure the effects we did not measure in this experiment, especially the ones we found suggested in the open-ended questions and the written and verbal feedback. It will also contain the old questions, so it will provide confirmation for the findings already presented here. The use of a different demographic composition will address some of the threats to validity discussed in the previous section.

6 Acknowledgements

References

1. Buttle, F., SERVQUAL: review, critique, research agenda, European Journal of Marketing, vol. 30, issue 1, MCB UP1996
2. Clements, P. and Northrop, L., Software product lines, 2001, Addison-Wesley
3. Cohn, M., User stories applied: For agile software development, 2004, Addison-Wesley
4. Dutoit, A.H. and McCall, R. and Mistrík, I. and Paech, B., Rationale management in software engineering, 2006, Springer
5. Ferrell, O.C. and Hartline, M, Marketing strategy, 2005, Thomson South-Western
6. Freedman, D. and Pisani, R. and Purves, R., Statistics, 2007, W.W. Norton & Co
7. Giesen, J., Volker, A., Requirements interdependencies and stakeholder preferences, Proceedings of the IEEE joint international conference on requirements engineering, IEEE 2002
8. Lauesen, S., User interface design: a software engineering perspective, 2005, Addison-Wesley
9. Martilla, J. and James, J., Importance-performance analysis The journal of marketing, JSTOR 1977
10. Moore, W., Concept testing, Journal of business research, vol.10, issue 3, Elsevier 1982
11. Oliver, R.L., Satisfaction: A behavioral perspective on the consumer, 2010, ME Sharpe Inc
12. Ruhe, G. and Saliu, M.O., The art and science of software release planning, IEEE software vol. 22, issue 6, 2005
13. Rupp, C., Requirements Engineering und -Management, 2009, Hanser

7 Appendix: Feature descriptions as user tasks

Task 1	Digitize receipt
Start:	The user has received one or more receipts
End:	The data from the receipts are archived
Subtasks	**Solution**
Import receipt	1 The system imports a picture of the receipt. 2 The system makes it easy to prepare the picture for recognition. 3 The system recognizes the text in the picture. 4 The system guesses a single tag and applies it to each expense item. The user doesn't predefine any tags.
Check and correct receipt data	5 The system allows the user to change any part of the recognized content in a receipt. 6 The system allows the user to change the tag for each expense item separately.
Archive data	7a) Option 1: The system archives the data on the servers of the system vendors (cloud storage). No local saving is possible. All data is saved automatically. 7b) Option 2: Instead of cloud storage, the system archives the data locally. The user has to trigger the saving. 8 The system offers export of the receipt data.

Task 2	View expenses report
Start:	The user needs information on expenses
End:	The relevant information has been viewed and possibly printed.
Subtasks	**Solution**
Select the input for the report	9 The system offers the user to select the receipts to be used in the report. 10 The system offers several search and filter options for finding relevant receipts in the receipt list. 11 The system offers templates for different types of report. 12 The system allows the user to input parameters for the report, e.g. a month for a report which shows expenses for a given month.
Produce report	13 The system processes the data needed for the report and shows onscreen a report in a print-friendly layout. 14 The system allows the user to print the report. 15 The system allows export of the report data.

9 Online Questionnaires

Online Questionnaires Programme

A Survey on Empirical Requirements Engineering Research Practices

Nelly Condori-Fernandez, Maya Daneva, Roel Wieringa
University of Twente,
Enschede, The Netherlands.
{n.condorifernandez, m.daneva, r.j.wieringa}@utwente.nl

Abstract. [Context and Motivation]. In recent years a number of checklists for empirical research in software engineering have been published. So far, the checklists for experimental research differ from those of observational research. This leaves the important commonalities between these kinds of research unexploited. Recently, a unified checklist has been published that identifies the commonalities and identifies the difference between these two types of research. So far, little work has been done to evaluate checklists in practice. **[Objective]** Our goal is to gain insight into the current practice of empirical research in requirements engineering. **[Method]** We surveyed the empirical research practice of participants of the REFSQ 2012 conference. The survey was part of the REFSQ 2012 Empirical Track. **[Conclusions]** We found that there are 15 commonly used practices out of the set of 27 that we took from the unified checklist. Second, we found that senior researchers and PhD students do not always converge in their perceptions about the usefulness of research practices. We discuss the import of these findings on RE research practice.

Keywords: empirical research checklist, survey, requirements engineering

1   Introduction

In recent years, there has been an increased interest in empirical research in Requirements Engineering (RE). This increase is not only reflected in the number of published empirical studies but also in the growth of methodological advise on empirical software engineering (SE). For example, we observe an increasing diversity of proposed checklists concerning the planning, execution and reporting on empirical SE studies [4][5][6][7]. The checklists for experimental research differ substantially from the checklist for observational case study research. This is a missed opportunity, because beyond the obvious difference that experimental research applies a treatment and observational research does not, there is considerable common structure among both kinds of research. Identifying this common structure enhances our understanding of the checklists and allows us to combine the best elements if both kinds of checklists into one. The unified checklist proposed by Wieringa [1] is based on an analysis of extant checklists as well as on checklists for empirical research [4][5][6][7] [8] as well as an earlier analysis of the logic of the engineering cycle [9].

After a first experimental evaluation [10], the checklist was simplified, and this simplified version was used as the basis for the current survey.

The majority of these checklists have not yet been sufficiently evaluated in terms of their usability and usefulness. What do we know about the use of these recommended practices in RE? And what do we know about the practice of empirical research in RE at all? We conducted a survey among the participants of the REFSQ 2012 conference to collect data about their empirical research practice by asking them, for each of the items of the unified checklist, whether or not this item belongs to their practice.

The paper is structured as follows. Section 2 describes the research method. Section 3 presents the survey results obtained and provides the discussion over these results. Finally, in section 4 we provide our final conclusions.

2　Method

With our survey we aim to address the following research questions:

RQ1: What are common practices in designing and reporting empirical research carried out by researchers and practitioners?

RQ2: What recommended practices reported in the literature do researchers and practitioners consider useful for designing and reporting empirical research?

We followed the guidelines of Kitchenham and Pfleeger in [3] to create a web-based survey, consisting of 50 questions (summarized in Table 1). 30 out of 50 questions were formulated to discover which of the recommended practices in the literature are performed by the respondents. Each of these questions was rated on a 3-point nominal scale ['yes', 'no', 'unsure I understand what you ask'].

The remaining questions were formulated in order to understand the usefulness perceived of the most recommended practices for empirical research (case studies and experiments). A 5-point Likert scale was used for this set of questions, where 1 = not useful and. 5 = very useful.

The questions focus on different recommended practices to be considered through six phases of the empirical cycle [2]: research problem investigation, research design, research design validation, execution and results evaluation. The questions were based on a revision of the unified checklist proposed by Wieringa [1]. We tested the questionnaire with 1 PhD student and 1 Post doc researcher, who have experience in designing experiments. The questionnaire testing discovered the unclear questions, and it helped us to remove some ambiguities.

Table 1. Summary of survey questions

	ID	Question	Scale
Research context	Q1	Is your empirical research usually motivated by the goal to improve some artefact ?	Nominal
	Q2	Do you usually define a top-level knowledge goal for your empirical research?	
	Q3	Do you usually review the current state of knowledge related to your empirical research?	
	Q4	Do you think that the following practices would be useful to have a better contextualization of your research?	Likert
	Q4.1	Definition of improvement goal	
	Q4.2	Definition of knowledge goal	
	Q4.3	Review of the current state of knowledge	
Research problem	Q5	Do you usually define a conceptual framework for the phenomena to be investigated in your research?	Nominal
	Q6	Do you usually operationalize the concepts of this framework?	
	Q7	Do you validate these operationalizations?	
	Q8	Do you usually formulate the research questions in your empirical research?	
	Q9	Do you usually describe the population in your empirical research?	
	Q10	Do you think that the following practices would be useful to improve the understanding of your research problem?	Likert
	Q10.1	Definition of relevant concepts of the phenomena to be investigated	
	Q10.2	Operationalization of the concepts defined	
	Q10.3	Validation of the operationalization of concepts	
	Q10.4	Formulation of research questions	
	Q10.5	Description of population	
Research design and justification	Q11	Do you usually justify the acquisition process of the object of study for your empirical research?	Nominal
	Q12	Do you consider any ethical issue in your research involving human subjects?	
	Q13	Do you usually justify the representativeness of the object of study for the population in your empirical research?	
	Q14	Do you usually consider all the assumptions of inference techniques to be used in your empirical research?	
	Q15	Do you usually plan the procedures to be followed in the experimental treatment?	
	Q16	Do you usually specify any instruments needed to apply the treatments of your experimental research?	
	Q17	Do you usually specify any instruments needed for measurement?	
	Q18	Do you usually specify procedures to be followed when performing measurements?	
	Q19	Could you indicate whether you usually consider the validity of the following issues?	
	Q19.1	Measures	
	Q19.2	Measurement procedure	
	Q19.3	Measurement instrument	
	Q19.4	Treatment	
	Q19.5	Treatment procedure	
	Q19.6	Treatment instrument	
	Q20	Do you think that the following practices would be useful to improve your research design?	Likert
	Q20.1	Justification of the acquisition process of the objects of study	
	Q20.2	Ethical issues	
	Q20.3	Representativeness of the objects of study selected	
	Q20.4	Consideration of all assumptions of the inference technique to be used	
	Q20.5	Specification of the treatments planning	
	Q20.6	Design of the instruments and procedures to apply the treatments	
	Q20.7	Design of the measurement instruments and procedures	
Execution	Q21	Do you think that is necessary to report what actually happened during the execution of an empirical research about the following issues?	Nominal
	Q21.1	Deviations from the acquisition plan of objects of study	
	Q21.2	Deviations from the treatment plan	
	Q21.3	Deviations from the measurement plan	
Results	Q22	Do you usually explain your observations in terms of underlying mechanisms or available theories?	Nominal
	Q23	Do you usually assess the plausibility of your explanations?	
	Q24	Do you usually answer the research questions explicitly?	
	Q25	Do you usually verify that the contributions to improvement goal are described in your report?	
	Q26	Do you usually verify that the contributions to knowledge goal are described in your report?	
	Q27	Do you think that the following practices would be useful to improve the report of your empirical results?	5-item Likert
	Q27.1	The use of mechanisms or available theories to explain your observations	
	Q27.2	Plausibility assessment of your explanation	
	Q27.3	Plausibility assessment of tested hypotheses	
	Q27.4	Contributions to improvement goal	
	Q27.5	Contributions to knowledge goal	

Moreover, in order to gather information about the respondents, five closed-ended questions were asked at the beginning of the survey. The information included the sector of their current job (e.g. academia); their role in the organization, experience years in requirements engineering, experience level in designing experiments or case studies. The survey was implemented using the Surveygizmo tool [12], and was configured to be accessible on laptops, tablets and mobile platforms.

2.1 Data collection

The survey was electronically distributed by the REFSQ 2012-participants mailing list, which was established to facilitate communication among the organizers of the conference, researchers and practitioners participating in the 18th International working conference on Requirements Engineering: Foundation for Software Quality [13]. From 110 participants that were registered at the REFSQ conference, 36 completed our survey, 6 participants answered partially and 7 participants abandoned the survey after reading the instructions. We collected survey data during two weeks, from 19 to 30 March 2012. Actually, the data collection was originally planned to be carried out only during the conference week, but with the purpose of increasing our response rate this was extended to one week more. Two reminder emails were sent to encourage participants who had not yet responded the survey to reply.

2.2 Respondents' characteristics

As is shown in Figure 1, the survey response captured a diverse of range of roles, since Master students from academia to Senior consultants from industry. 17 out of 42 respondents were PhD candidate (40,5%), only one of them worked also in the industry sector. The other almost half of respondents were senior researchers (42,9%), where 15of them come from academia, 2 from industry and 1 from both sectors.

The survey participants also reflect a diverse range of experience with requirements engineering (See Figure 2) and empirical research (See Table 2).
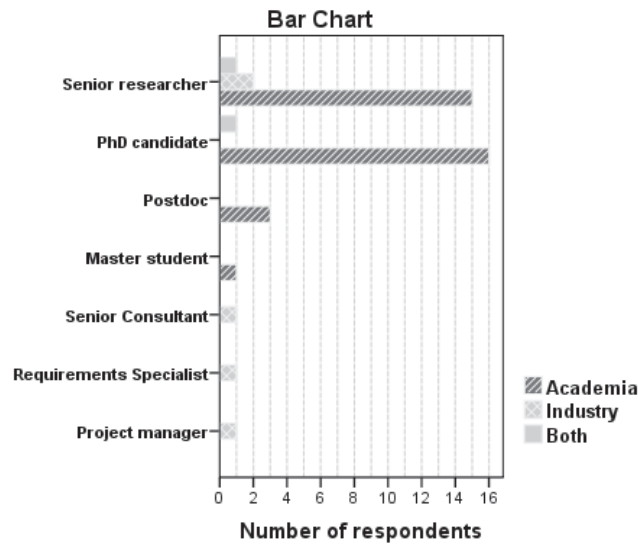
Figure 1. Distribution of respondents per role in their current organization
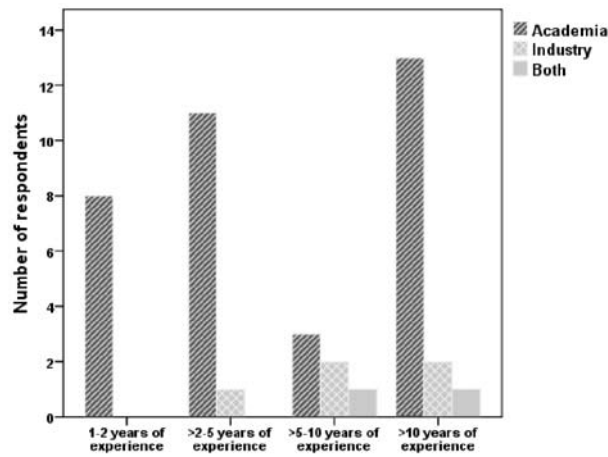
Figure 2. Distribution of respondents' experience with Requirements Engineering

Table 2. Experience in designing experiments or cases studies.

Number of times		Sector				Number of times		Sector			
		Academia	Industry	Both	Total			Academia	Industry	Both	Total
designing experiments	>30	0	0	0	0	designing case studies	>30	1	1	0	2
	>20-30	0	1	0	1		>20-30	2	1	0	3
	>10-20	4	0	0	4		>10-20	4	1	1	6
	>5-10	6	1	1	8		>5-10	3	1	0	4
	1-5.	21	2	1	24		1-5	21	0	1	22
	0	4	1	0	5		0	4	1	0	5
Total		35	5	2	42	Total		35	5	2	42

3 Survey results

The complete report contains the Chi-square statistics for the survey [11].

3.1 Research context.

As is shown in Figure 3, 35 out of 39 respondents (89%) acknowledge that they usually review the current knowledge related to their empirical research (Q3). 32 of them (82%) stated that they usually define a knowledge goal when investigating an engineering problem (Q2). It is important to remark that 6 respondents did not get to understand this question. 3 out of these 6 respondents were post-Docs, 2 PhD students, and 1 a senior researcher. However, 34% out of 39 responses stated that they omit the definition of improvement goals in their empirical research (Q1) as part of their practice. Only 1 respondent reported the question as not understandable. This respondent was a senior researcher with a medium level of empirical experience.
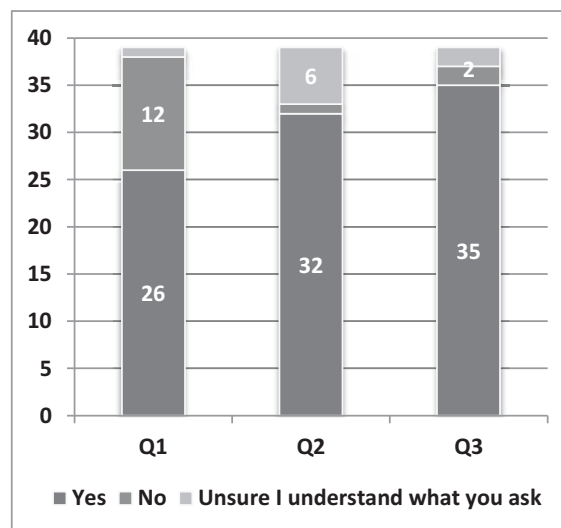
Figure 3. Distribution of practices on contextualization of empirical research problems

Applying the Chi-square test of goodness of fit, we found that the definition of improvement goal (Q1) can be considered as a common practice but only among senior researchers (p=0,004). However, for the definition of knowledge goal (Q2) and review of the current state of empirical knowledge (Q3), we corroborated enough evidence to consider them to be common practices among PhD students and senior researchers (p=0,001).

Table 3 shows that the percentage of neutral responses was higher for the first recommended practice "definition of improvement goal" than for the other two practices. This means that 23% of respondents preferred to choose a neutral position. In general terms, respondents tend to perceive the last two practices as very useful (above 50%).

Table 3. Perceived usefulness of the recommended practices for contextualizing

	Perceived Usefulness				
Question	**1 (not useful)**	**2**	**3 (neutral)**	**4**	**5 (very useful)**
Q4.1	2.9%	0.0%	22.9%	34.3%	40.0%
Q4.2	0.0%	2.9%	14.3%	25.7%	57.1%
Q4.3	0.0%	0.0%	11.1%	27.8%	61.1%

3.2   Research problem. Figure 4 shows our observations collected from the next five questions(Q5-Q9); where we can note that the practice with highest percentage of respondents (97%) is the "formulation of research questions" (Q8), followed surprisingly by the "description of the population to be investigated" practice (Q9) with a 89% of respondents. We also noted that only 57% of respondents recognized to the "definition of relevant concepts of the phenomena to be investigated" (Q5) as part of their common practices. The other half of respondents stated that they did not consider this practice in their empirical studies (22%) or simply were not able to understand the question (18%). Figure 4 also illustrates that the total of affirmative responses for question Q6 and Q7 decrease drastically. This is because the Q6 and Q7 were enabled only if respondents answered the respective previous question (Q5 and Q6) affirmatively. Thus, only 23% indicated that the validation of the most relevant concepts previously operationalized is considered in their empirical research.
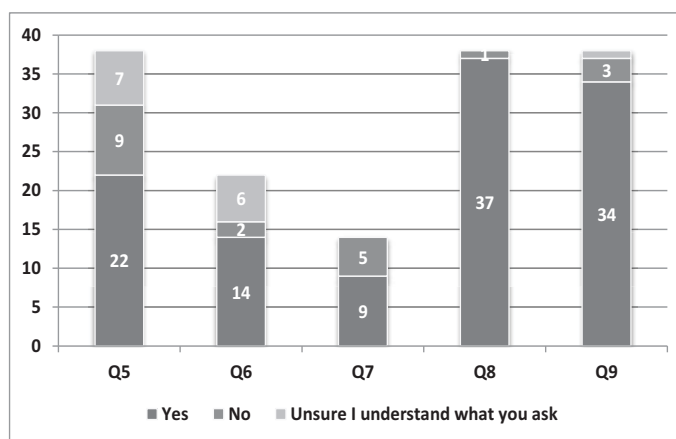
Figure 4. Practices applied to enable a better understanding of a research problem

Applying the Chi-square test for this set of questions, we found enough evidence only for the last two practices recommended for understanding better the problem to be investigated empirically: formulation of research questions and description of population. ($p < 0,05$).

Analyzing the distribution of frequencies for usefulness perceived (Table 4), we can see that only 38.7% of respondents perceived the practice "operationalization of the relevant concepts" as very useful, while 26% chose a neutral response.

We also noted that although the "description of population" practice was considered as a common practice by the senior researchers and PhD students, only 51% of respondents perceived this practice as very useful and 32% as useful. A possible explanation could be that majority of our respondents were more familiarized with case studies, where concepts on population and operationalization are not sufficiently addressed by respondents.

Table 4. Perceived usefulness of the practices recommended for understanding the research problem

| Question	Perceived Usefulness				
	1 (not useful)	2	3 (neutral)	4	5 (very useful)
Q10.1	0.0%	2.9%	11.4%	25.7%	60.0%
Q10.2	0.0%	6.5%	25.8%	29.0%	38.7%
Q10.3	0.0%	3.0%	27.3%	21.2%	48.5%
Q10.4	0.0%	0.0%	2.7%	18.9%	78.4%
Q10.5	0.0%	2.7%	13.5%	32.4%	51.4%

3.3 Research design and justification. In this section, we report our results collected from the questions (Q11-Q19.6) formulated in order to know which of the practices are most applied by the respondents for getting better research designs and justifications. Figure 5 shows that the practice of "justifying the acquisition process of the object of study" is the one that is least applied by the respondents (48%); followed by the practice of "considering all assumptions of inference techniques" (17 out of 37). In both cases, a considerable number of respondents found difficulties to understand these questions (Q11 and Q14). This can be due to the fact that the questions were rather ambiguous, or that respondents are not familiarized with the terminology, precisely because these recommended practices are not applied by them.

We also noted that 35% of respondents did not consider any ethical issue in their empirical research (Q12). This observation can be due to the fact that respondents are partially aware of the meaning of ethics (e.g. they can believe that ethical issues should only be considered where experiments could induce life threatening conditions in humans).

On the other hand, considering that questions Q15 and Q16 showed only whether the respondents had experience in designing experiments, we noted that 3 out of 4

respondents, who did not understand the question Q16, were senior researchers with a high level of empirical experience. However, 10 of 28 respondents who stated that they consider this practice ("specification of any instrument to apply the treatments"), were also researchers with a high level of empirical experience.

Applying the chi-square test, we found that although 28 respondents answered affirmatively to the question Q16; there is only a significant difference in the opinions given by PhD students ($p=0,001$) but not by senior researchers ($p=0,02$). For questions Q13 (justification of the representativeness of the object of study for the population), Q17 (specification of any instrument for measurement), and Q18 (specification of procedures to be followed when performing measurements), we found enough evidence to affirm that these three practices are those most applied by our respondents.
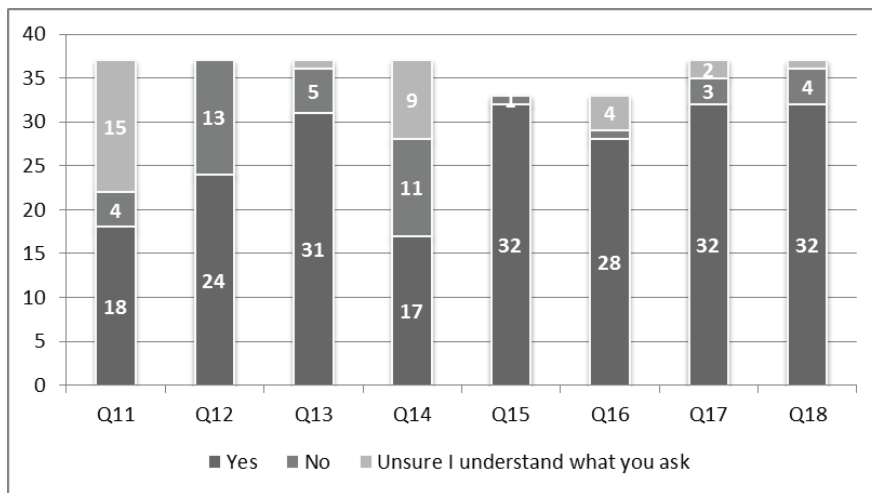
Figure 5. Practices applied to get a better research design and justification (part I)

Figure 6 shows results about the practices recommended regarding the validity of measures (Q19.1), measurement procedures (Q19.2), measurement instruments (Q19.3), treatments (Q19.4), treatment procedures (Q19.5) and treatment instruments (Q19.6). More than 70% of respondents stated that they apply the first four practices in their research. However, we corroborated that the last two practices recommended are only applied by senior researchers.
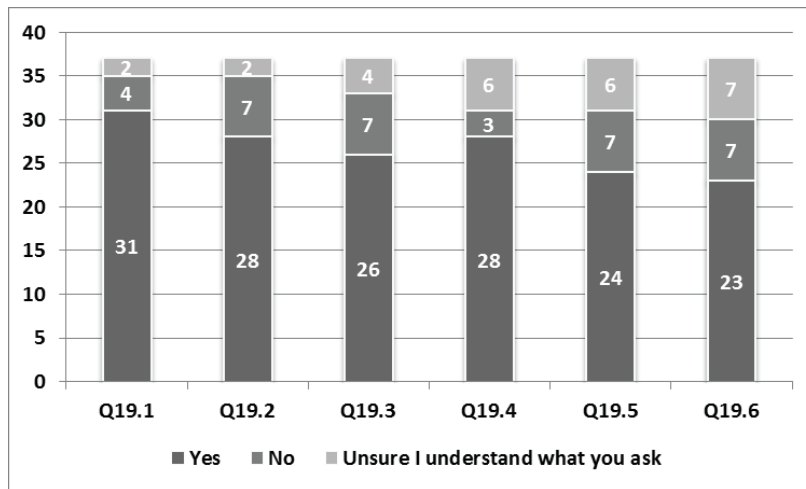
Figure 6. Practices applied to enable better research design and justification (part II)

We can see that only 16.7% of respondents perceived the practice "justification of the acquisition of the object study" as very useful, while 30% chose a neutral response.

We also noted that although the "specification of measurement instruments and procedures" practices were considered as a common practice by the senior researchers and PhD students, only 47% of them perceived both practices as very useful and 23.5% preferred to choose a neutral response. Once, this could be due to that majority of our respondents were more familiarized with case studies, where measurement concepts are less used than by researchers familiarized with experiments.

Table 5. Perceived usefulness of the practices recommended for research design and justification

Question	Perceived Usefulness				
	1 (not useful)	2	3 (neutral)	4	5 (very useful)
Q20.1	6.7%	13.3%	30.0%	33.3%	16.7%
Q20.2	8.8%	29.4%	20.6%	11.8%	29.4%
Q20.3	0.0%	3.0%	18.2%	30.3%	48.5%
Q20.4	0.0%	12.5%	18.8%	28.1%	40.6%
Q20.5	0.0%	9.1%	24.2%	30.3%	36.4%
Q20.6	6.3%	9.4%	25.0%	18.8%	40.6%
Q20.7	5.9%	2.9%	23.5%	20.6%	47.1%

3.4 Research execution

Concerning the questions on research execution, the respondents mostly declared that they understand the questions. However, it is noteworthy that in Q21.1, about the report of deviations from acquisition plan of objects study, there were a higher number of subjects who were unsure about the meaning of this practice in comparison to other questions in this section (see Figure 7).

Overall, these answers suggest that nearly 90% of the participants do consider it necessary to report what actually happened during the execution of empirical research, in terms of deviations from either the acquisition plan of objects of study (Q21.1), or the treatment plan (Q21.2), or the measurement plan (Q21.3).

Applying the chi-square test, we found that although 26 respondents answered affirmatively to the question Q21.1; there is only enough evidence to confirm that "the report of deviations from the acquisition plan of objects of study" is a common practice among PhD students (p=0,002) but not by senior researchers (p=0,041). However, reporting the deviations from the treatment and measurement plans are considered valuable information to be reported (by senior researchers and PhD students).
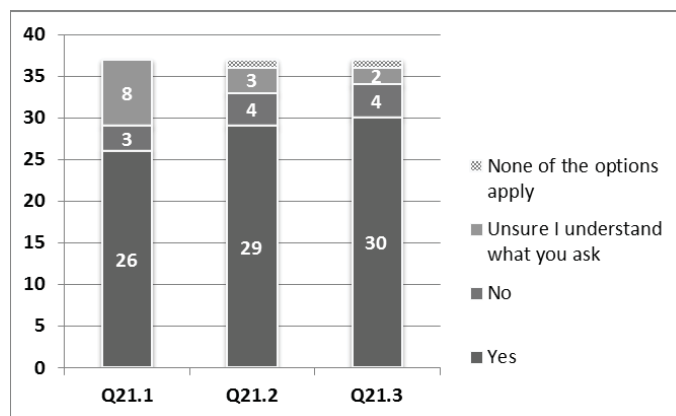

Figure 7. Research execution practices

3.5 Results analysis.

Questions Q22 through Q26 concern what the participants say that they do when analyzing their results (see Figure 8).

Regarding the terminology used, everyone understood the question Q24, but a few respondents answered that they were unsure about the meaning of "explain observations in terms of underlying mechanisms or available theories" (Q22), or "assess the plausibility of explanations" (Q23), or "verify that contributions to the improvement/knowledge goal are described" (Q25 and Q26).

According to what people usually do in their analyses, we can say that nearly 90% of the participants try to answer the research questions explicitly. In contrast, about 22% of the participants (majority of them PhD Students) affirmed that they do not usually explain their observations in terms of available theories (Q22), which suggests that

these researchers follow a more descriptive analysis, simply reporting their observations without making the effort to link it with underlying mechanisms. Applying the chi-square test, we corroborated that the first two practices (Q22 and Q23) are usually applied by senior researchers (p=0,004) but not by PhD students (p=0,04).
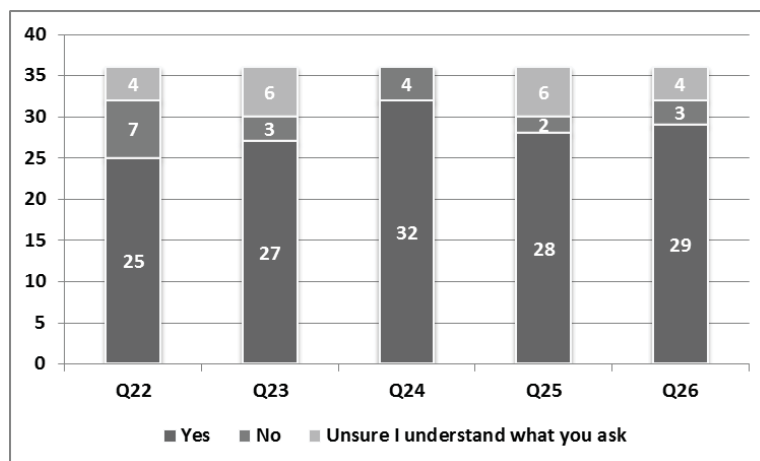
Figure 8. Result analysis practices

Prior questions dealt with what researchers do commonly when they analyze their results. However, it is also interesting to know more about the perceived usefulness on the recommended practices included in this section. Table 6 shows the results on a 5-point Likert scale of the perceived usefulness for the practices Q27.1-Q27.5. The results show that the participants mostly consider useful or very useful all the practices recommended in order to improve result analysis.

Table 6. Perceived usefulness of practices recommended for obtaining better empirical reports

Question	Perceived Usefulness				
	1 (not useful)	2	3 (neutral)	4	5 (very useful)
Q27.1	2.8%	0.0%	5.6%	30.6%	61.1%
Q27.2	2.9%	0.0%	17.6%	23.5%	55.9%
Q27.3	0.0%	3.0%	9.1%	21.2%	66.7%
Q27.4	6.1%	0.0%	9.1%	27.3%	57.6%
Q27.5	0.0%	0.0%	6.1%	36.4%	57.6%

4 Summary and Conclusions

Context of empirical research: The definition of improvement goals appears to be a common practice but only among senior researchers. A possible generalization is that the ability to put research in a wider, practical context, tends to be absent in PhD researchers but grows with experience. On the other hand, the definition of knowledge goals and review of the current state of empirical knowledge is common practices among both senior researchers and PhD students. Our respondents perceived these recommended practices as useful.

The research problem: Our respondents do formulate research questions and describe the population to be investigated. Definition, operationalization and validation of a conceptual framework was less widely practiced. About 26% of our respondents did not perceive operationalization or validation of concepts as useful. This could indicate a lack of theory usage in RE research. A possible explanation is that these two practices are not currently required for publishing empirical research.

Research design and justification. Most of our respondents justified the representativeness of the object of study, and specified and validated measurement instruments and procedures. Justification of ethical issues and justification of inferences techniques were not widely practiced.

Research execution. PhD students agreed that it is necessary to report what actually happened during research. Some of the senior researchers did not understand what was meant. We have no explanation for this.

Results analysis. The majority of PhD Students do not usually explain their observations in terms of underlying mechanisms or available theories.

Overall, respondents tended to give greater importance to practices in results analysis than to practices in research design. Since analysis must be based on proper design, this points to an important improvement possibility of empirical RE research practice.

5 Acknowledgments

References

1. Wieringa, R.J. (2012) A Unified Checklist for Observational and Experimental Research in Software Engineering (Version 1). TR-CTIT-12-07, CTIT, UT, Enschede. ISSN 1381-3625
2. Wieringa R. J. Design science as nested problem solving. ACM 4 the DESRIST, 2009 , pp. 1–12.
3. Kitchenham, B.A., Pfleeger, S.L.: Principles of Survey Research - Part 3: Constructing a Survey Instrument. SIGSOFT Software Engineering Notes 27, 20–24 (March 2002).

4. Pfleeger S., "Experimental design and analysis in software engineering," Annals of Software Engineering, vol. 1, no. 1, pp. 219–253, 1995.
5. Kitchenham B. A., Pfleeger Sh. L., Pickard L. M., Jones P. W., Hoaglin D. C., El Emam K., and Rosenberg J. Preliminary guidelines for empirical research in software engineering. IEEE Trans. Softw. Eng. 28(8), 2002, pp. 721-734.
6. Jedlitschka A., Pfahl D., "Reporting guidelines for controlled experiments in software engineering," IEEE ISESE 2005, pp.94-104.
7. Runeson P. and Höst M. 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Softw. Engineering. 14(2), 2009, pp. 131-164.
8. Moher D., Hopewell S., Schulz K., Montori V., Gøtzsche P.C., Devereaux P.J., Elbourne D., Egger M., and Altman D.G. (for the CONSORT Group). CONSORT 2010 Explanation and Elaboration: updated guidelines for reporting parallel group randomised trial. British Medical Journal 2010, pp. 340-c869.
9. Wieringa, R.J. and Maiden, N.A.M. and Mead, N. and Rolland, C. (2006) Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. Requirements Engineering, 11 (1). pp. 102-107.
10. Condori-Fernandez N., Wieringa R., Daneva M., Mutschler B., Pastor O. Evaluation of a checklist for designing empirical research and reporting about it. March 2012. Submitted.
11. Condori-Fernandez N., Daneva M., and Wieringa R. A survey on empirical requirements engineering research practices. TR-CTIT-12-10, CTIT, UT, Enschede. ISSN 1381-3625
12. Surveygizmo. Online Survey Software & Questionnarie tool (2005). Retrieved 24 April 2012, Web site: http://www.surveygizmo.com/
13. The Ruhr Institute for Software Technology, Universität Duisburg-Essen, 18th International conference on Requirements Engineering: Foundation for software quality (2012). Retrieved 24 April, 2012, Web site: http://www.refsq.org/2012/

Preliminary Results of a Survey on Requirements Engineering for variability-intensive Software Systems

Christian Manteuffel, Matthias Galster, Paris Avgeriou

University of Groningen, The Netherlands
c.manteuffel@student.rug.nl, m.r.galster@rug.nl, paris@cs.rug.nl

Abstract. Context: The success of requirements engineering (RE) methods depends, among others, on the domain and characteristics of the domain in which they are applied. One particular domain is the domain of variability-intensive systems. Objective: Variability affects the whole software development process, including RE. Thus, we aim at understanding RE in the context of variability-intensive systems. Method: We conducted a survey at REFSQ 2012 using an online questionnaire. Results: We analyzed data from 23 respondents. We found that even though respondents consider variability important, there are still many issues when dealing with it. Also, current RE methods seem not appropriate to properly handle variability during RE. Conclusions: New approaches to support handling variability should not only focus on one RE activity, but cover all RE activities. Furthermore, to better understand RE in the context of variability-intensive systems, more industrial software projects should be studied, beyond general surveys with practitioners.

1 Motivation

Variability is the ability of a software system or artifact to be changed (e.g., extended, customized or configured) for use in a specific context. To enable variability, certain parts of the system are not fully defined during early iterations, but later when, for instance, more details about a concrete customer are known. For example, during early iterations we might identify what parts of a system should be variable (e.g., in terms of "variation points") and what the options are to resolve variability (e.g., in terms of "variants"). Later, we decide how to actually resolve this variability and what variant to choose. This means, variability can be interpreted as planned or anticipated change, or as "planned" requirements uncertainty.

Variability imposes challenges on the whole software development process, including requirements engineering (RE) as it affects functional and non-functional requirements. Variability is an important concern during RE and a key fact of most, if not all, systems [7]. Examples of variability-intensive systems include systems for which design decisions are deferred to when more details about concrete customer requirements are known, highly configurable single systems, systems that support multiple deployment / operation and / or maintenance scenarios, product lines, open

platforms, self-adaptive systems, or service-based systems that support dynamic runtime composition of web services [16].

RE in the context of variability has mainly been addressed in the domain of software product lines [1]. For example, industrial challenges with variability management in product lines (e.g., handling complexity or extracting required variability) have been identified [3]. General RE challenges in the context of large-scale systems were studied in [12], but no industrial studies on RE in the context of variability-intensive systems exist. Furthermore, existing RE methods lack the ability to anticipate change [13]. In RE, change is often anticipated after documentation [15] rather than as part of requirements elicitation or analysis. To reduce the impact of variability on different parts of a software system and on different software development artifacts, change should be anticipated during requirements elicitation [13]. Finally, there are no empirical studies on the applicability of existing approaches in the context of RE that anticipate change in real-world projects [13]. In particular, an understanding of challenges that occur during RE for variability-intensive industrial systems is missing. Therefore, this paper reports results of a survey conducted at REFSQ 2012 that aimed at understanding RE in the context of variability-intensive software systems. The survey included participants from industry and academia.

The rest of this paper is organized as follows. In Section 2 we discuss related work. Section 3 presents the design of the survey. In Section 4 we present the results and Section 5 acknowledges threats to validity. The paper ends with conclusions and directions for future work in Section 6.

2   Related Work

Chen et al. performed a systematic literature review on variability management approaches [4]. They found that the majority of research on variability management in product lines covers variability during RE but in general existing approaches for variability management lack validation [4]. Another systematic literature review assessed the current state and quality of research on RE for product lines. It found that current studies are limited in terms of validity, and lack sufficient guidance for practitioners, which limits the use in industrial settings. The review concludes that more effort should be invested in tool support and guidance to adopt methods in practice [19].

Furthermore, challenges and issues with variability have been identified. While Chen et al. separate technical challenges (e.g., handling complexity) and non-technical challenges (e.g., people and mindset) [3], Jaring and Bosch identified three major issues: variability identification, variability dependencies and the lack of tool support [8].

Nolan et al. evaluated the impact of unmanaged requirements uncertainties in later phases of a software project. The authors discuss a method to analyze requirements uncertainties during RE. They found that for a "traditional non-product line project, over 80% of requirements uncertainty can be predicted at project launch" [14].

3 Design of the Study

3.1 Goal

The goal of the survey is to contribute to an understanding of RE in the context of variability-intensive systems. Therefore, we conduct a survey to collect information from practitioners and researchers on the role of variability during RE and how variability is handled. Based on the study goal, we define four research questions:

RQ1 How important do practitioners and researchers consider variability as a concern during RE for variability-intensive systems?

RQ2 What concerns do requirements engineers have with regard to handling variable requirements during RE?

RQ3 Which RE activities are affected most by variability?

RQ4 What methods are most promising to handle variable requirements during RE?

We ask RQ1 because variability is often one concern among many other concerns (e.g., other non-functional requirements). Thus, we are interested in finding out how variability is perceived in comparison to other concerns. This question is of interest for researchers and practitioners.

We ask RQ2 because requirements engineers face difficulties when dealing with variability. We are interested in finding out what these difficulties are with regard to RE. This helps researchers develop new approaches to support RE practitioners. By asking this question also to researchers we can complement our results obtained from practitioners: Researchers usually have a broad overview of the state-of-the-art in a certain area and therefore might have different insights into problems with handling variability.

We ask RQ3 because variability has an impact on the whole software development process, including RE. To get a more detailed understanding of the effect of variability on RE, we study how variability affects requirements elicitation, analysis and negotiation, documentation, validation, etc. This question is interesting for researchers and practitioners.

We ask RQ4 because we want to identify good practices for RE for variability-intensive systems. This question is interesting for practitioners who want to reuse good practices, but also for researchers to understand the current state-of-practice. Researchers often have an overview of existing RE methods to handle variability. Thus, this question also helps identify differences in the perceptions of researchers and practitioners.

3.2 Subjects and Sampling

The population under study are RE practitioners and researchers. The population is restricted to respondents with experience in the field of variability and RE. This means, subjects were required to have research or industry experience with

variability-related topics, such as product lines, reference architectures, or self-adapting systems. Given that the study was conducted at REFSQ, the population was not limited in terms of years of experience. To find participants we used purposive sampling [6] and encouraged attendees of REFSQ 2012 to participate in the survey.

In total, 24 responses were collected, out of which one was incomplete and thus was excluded from the results. Of the remaining 23 responses, 17 participants had experience in the software industry. On average the participants had eight years of industrial experience (minimum 0.5 years and maximum 45 years). Twenty-two participants had experience in academic research. On average, participants spent 7.5 years on RE research and 2.3 years on research related to variability. Twenty-one participants had received some sort of training related to RE and 14 participants had received training related to handling variability throughout their career. Out of 23 participants, eight answered survey questions from a practitioner's perspective, the rest from a researcher's perspective. In contrast to the researcher's perspective, the practitioner's perspective included questions about the organization in which respondents were working (see Section 3.3).

3.3 Data Collection

Data were collected through a self-administered online questionnaire. The reason for using an online questionnaire was that multiple subjects could answer the questionnaire concurrently. Furthermore, a self-administered questionnaire has the advantage that subjects and researchers do not have to synchronize time and place and that participants could fill in the questionnaire independent from researchers. Also, an online questionnaire avoids introducing errors in data that could occur when manually entering data into computer systems from paper-based questionnaires. In order to avoid poorly phrased questions, we piloted the questionnaire with subjects from the target population. We revised the questions based on their feedback and comments.

The questionnaire included structured questions and unstructured questions. Structured questions could be answered using Likert-scale or pre-defined answer options [9], unstructured questions allowed numeric answers or free text. For some questions, more than one answer could have been applicable (e.g., role of participants). In this case, participants could choose the answer that describes their role, etc. best. Furthermore, for most questions we allowed participants to provide additional comments to complement their answer, and to skip questions if they did not feel comfortable answering it. The actual questions of the questionnaire are provided in Section 4 when we discuss the results.

As we addressed both practitioners and researchers, we added an additional set of questions about the organization of participants that answered questions from a practitioner's perspective. For example, we asked questions about the domain, company size, etc. This was to ensure that we covered a broad range of industrial experiences. Practitioners worked in small to large organizations and had various roles, including project managers, requirements engineers, etc. Furthermore, industrial participants came from many different domains. However, as only eight participants answered the questions from a practitioner's perspective, we were not able to determine relationships between company size, domain, role of participants, etc. and

the answers participants from industry gave. Practitioners were asked to answer the questions based on their practical experience. This resulted in two paths, which were controlled by a check-question to find out if a participant was from industry or research. The structure of the questionnaire as well as its relation to our research questions is illustrated in Figure 1.
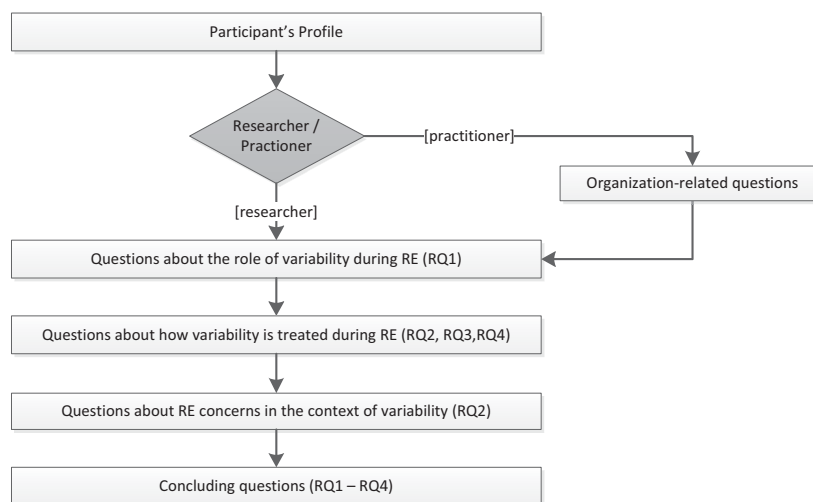
Fig. 1. Flow of questions in the questionnaire.

The questionnaire was available from March 19, 2012 to March 30, 2012. This includes the time of the REFSQ conference plus one week after the conference. During the conference 18 people participated in the survey. In the week after the conference, six people filled in the questionnaire.

4 Analysis

We used descriptive statistics to analyze the data [10]. The diagrams in this section show a stacked bar representing the percentage or the absolute number of answers to the respective questions. The label on the bars shows the absolute number of answers according to the participant's background.

4.1 RQ1 – Importance of Variability

We asked participants about the importance of variability in the context of RE. Answers were provided in a three-point scale. As illustrated in Figure 2, more than 70% of participants reported that variability is very important in the context of RE.

Less than 30% of the participants rated variability moderately important, while no respondent considered variability as unimportant. The answers show a clear trend that variability is an important concern, especially for participants with a background in research.
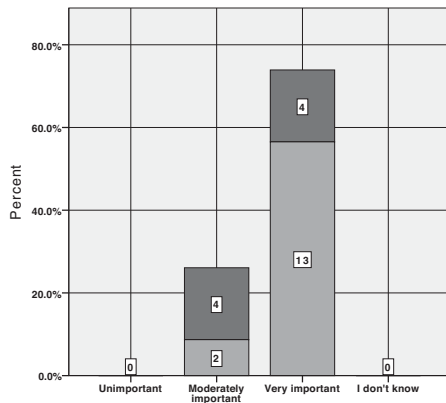
Fig. 2. Frequencies of answers to question "How important do you think is variability in the context of RE".
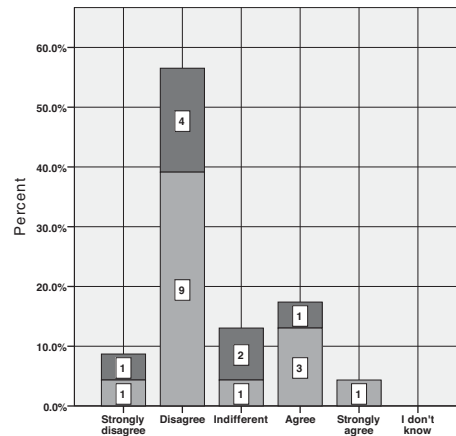
Fig. 3. Frequencies of answers to question "Based on your experience, how would you rate the following statement: During RE variability is usually strictly and explicitly managed."

We also asked if participants agree or disagree with the statement that variability is usually strictly and explicitly managed during RE. The majority of participants disagreed or strongly disagreed (65%) with the statement (Figure 3). Three participants were indifferent (13%), while five participants agreed with the statement (21%). There is no major difference between responses of participants from academia and participants from industry.

Summary of RQ1: Based on the responses we obtained from REFSQ participants, we conclude that even though variability is considered important, it is not sufficiently handled during RE.

4.2 RQ2 – Concerns regarding Variability

For the second research question, we asked participants about challenges and issues with respect to handling variability during RE. Participants were asked to select multiple requirement characteristics (e.g. modifiability, completeness) that are more difficult to achieve in the presence of many variable requirements. As depicted in Figure 4, consistency was selected most frequently (>80%), followed by traceability and completeness (both > 50%). All other characteristics were mentioned by less than 50% of the participants. There is no noteworthy difference between academic

researchers and practitioners, except for testability, which has been selected by ten researchers but only by one practitioner. This could be an indication that testability is more of an academic problem rather than a problem faced by practitioners in real projects.
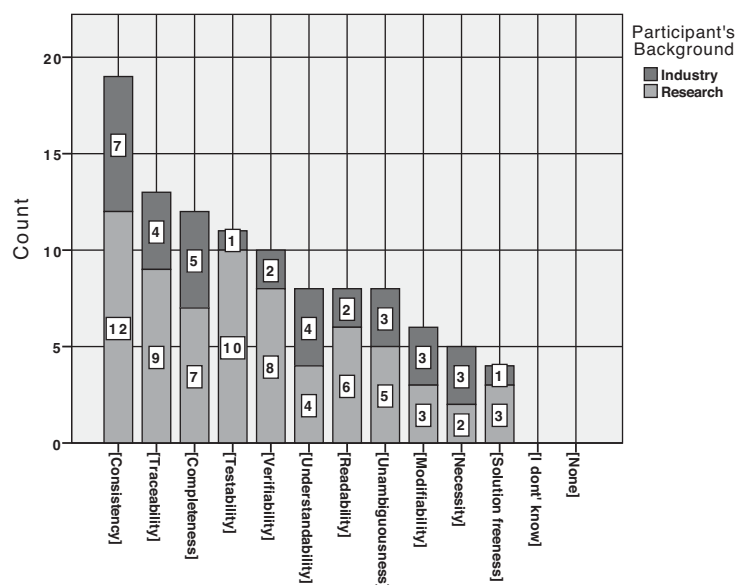
Fig. 4. Frequencies of the answers to question "Which requirements characteristics do you think are more difficult to achieve in the presence of many variable requirements?"

Furthermore, we asked about the most pressing technical issues when dealing with variability. This question also allowed multiple responses. As illustrated in Figure 5, the majority of participants (56%) mentioned handling complexity as most pressing technical issue. Ten out of 23 participants selected consistency management, extracting variability, knowledge harvest and management, and identifying commonalities and variabilities.

We asked the same question for non-technical issues, as illustrated in Figure 6. The two most mentioned issues are mindset-change (65%) and people (61%). The three other options were selected by less than 35%.

Furthermore, we asked the participants about the sufficiency of validation methods to validate variability in requirements, their satisfaction with current ways to specify variability in requirements, and the capability of RE methods to deal with variability. The answers to these questions do not show a clear trend (Figure 7 and 9). However, Figure 8 shows that the majority of participants agreed that ways of specifying variability are often unsatisfactory.

Summary of RQ2: Concerns related to handling variable requirements during RE include a) ensuring consistency of requirements despite variability, b) coping with complexity, c) ensure the right mindset of people involved, and d) have methods available to support specifying variability in requirements.
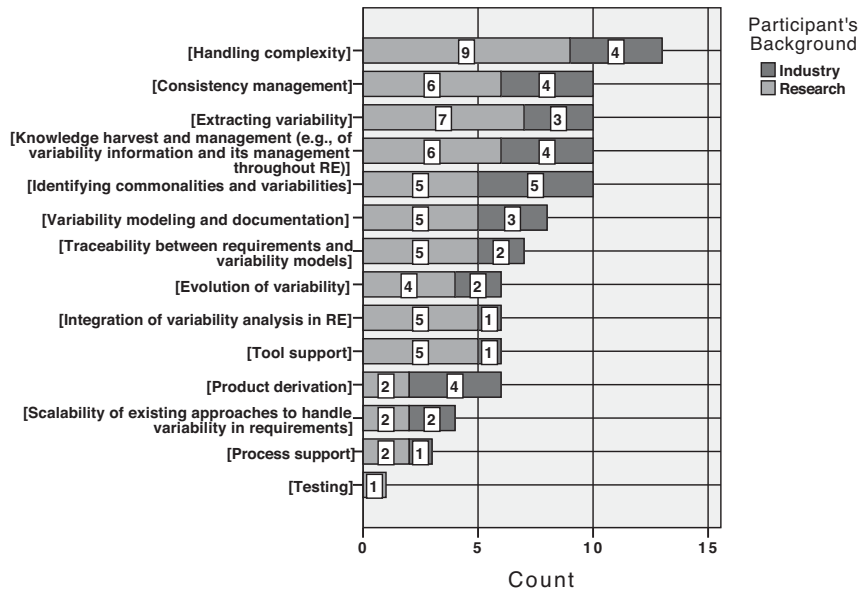
Fig. 5. Frequencies of the answers to question "What do you think are the most pressing technical issues when dealing with variability?"
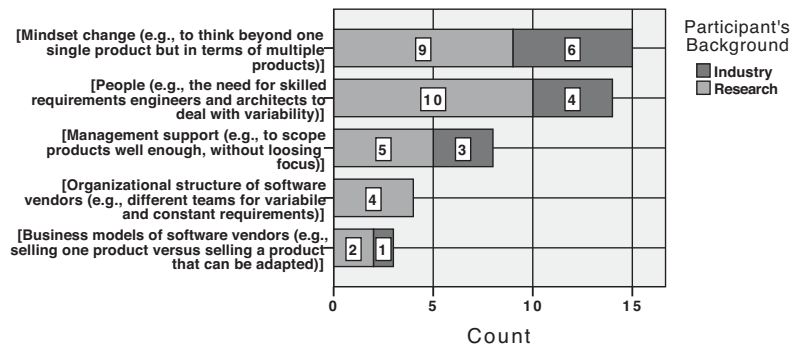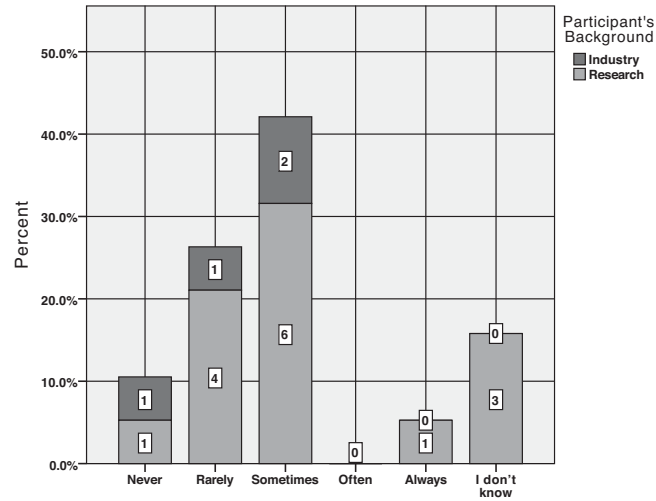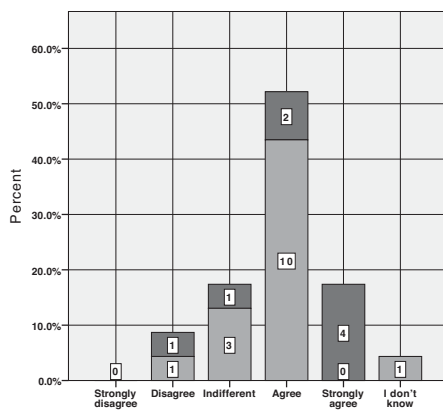

Fig. 6. Frequencies of answers to question "What do you think are the most pressing non-technical issues when dealing with variability during RE?"
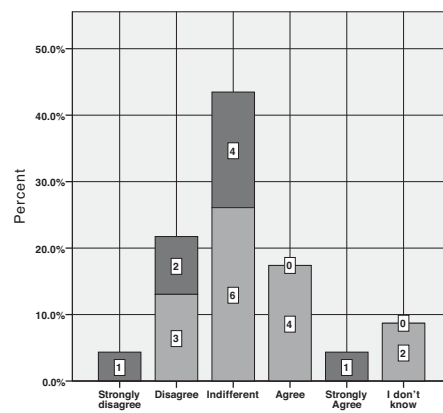
Fig. 7. Frequencies of the answers to question "Based on your experience, how would you rate the following statement: Current validation methods are sufficient to validate variability requirements during RE."

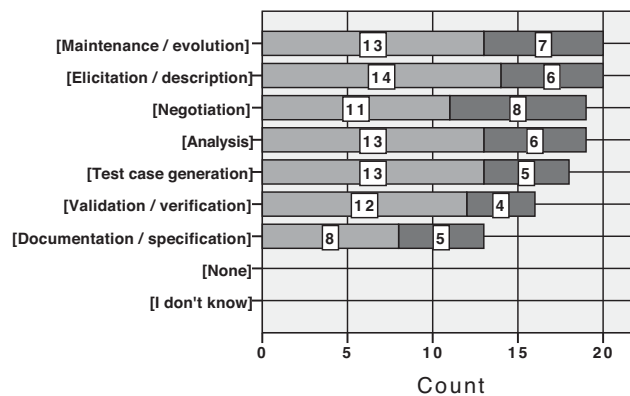**Fig. 8.** Frequencies of the answers to question "Based on your experience, how would you rate the following statement: Ways of specifying variability in requirements are often unsatisfactory."

Fig. 9. Frequencies of the answers to question "Based on your experience, how would you rate the following statement: Existing RE methods are capable of dealing with variability and variability in requirements"

4.3   RQ3 – RE Activities affected by Variability

Figure 10 shows the frequencies related to identifying RE activities that are affected by variability in requirements. The answers show that all RE activities are affected. Each activity has been selected by more than 50% of the participants with a high evidence for maintenance and elicitation (> 80%). In summary this means that there is not one single RE activity that requires special attention when handling variability.

Fig. 10. Frequencies of the answers for question "What RE activities are affected by variability in requirements?"

4.4   RQ4 – Methods promising to handle Variability

All questions regarding RQ4 were multiple-choice. As illustrated in Figure 11, Goal-oriented RE is considered to be most promising for handling variability in requirements (60%), followed by use-case-based RE (35%). The answers show a tendency towards goal models as preferred modeling language (40%, Figure 12). As goal models are widely used in RE, many researchers and practitioners are familiar with them. This could be an explanation why documentation and specification are not significantly affected by variability (see Section 4.3). Figure 12 also shows that none of the modeling languages is favored by the majority of participants.

Figure 13 shows that 52% of the participants consider requirements inspections as applicable to check variability requirements. Informal desk checks were selected in nine out of 23 cases. However, if practitioners and researchers are analyzed individually, informal desk checks were selected by 62.5% of the practitioners while requirement inspections were chosen by 37.5%.

Summary of RQ4: Goal-oriented requirements engineering including goal models to describe variability, and requirements inspections including desk checks for requirements verification and validation seem to be a good practice to handle variable requirements during RE.
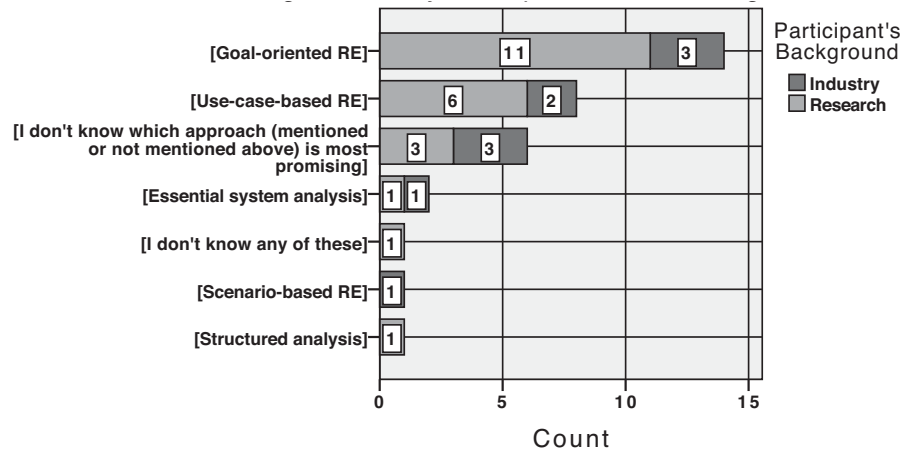
Fig. 11. Frequencies of the answers for question "What RE analysis approaches do you consider most promising for handling variability in requirements during RE?"
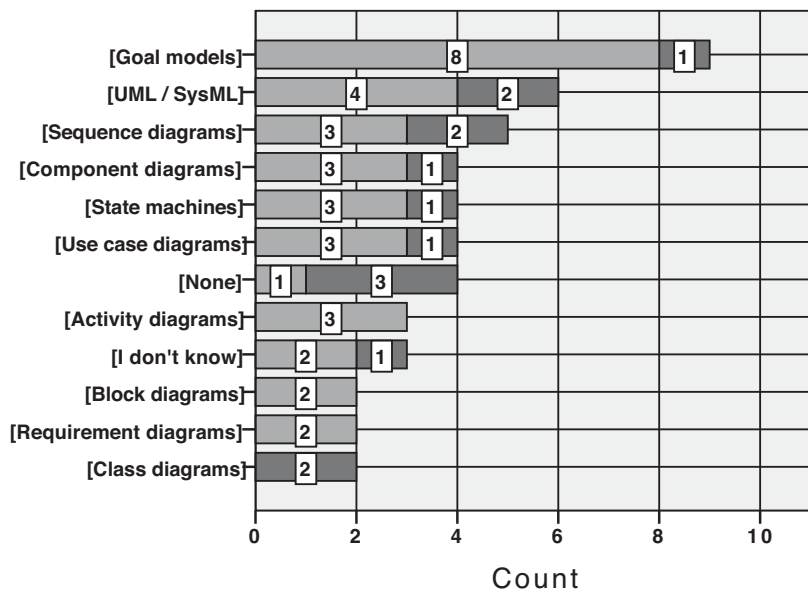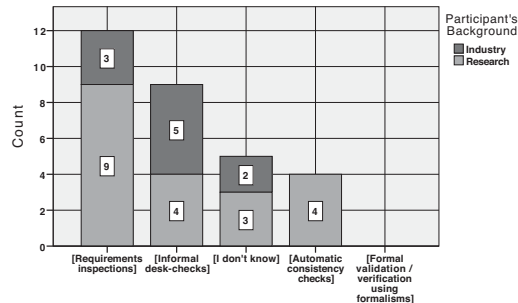

Fig. 12. Frequencies of the answers for question "What modeling languages are usually used to model variability in requirements?"

Fig. 13. Frequencies of the answer for question "What requirements validation and verification methods can be applied to check variability requirements?"

5 Threats to Validity

Internal validity: Internal validity is about confounding variables and other sources that could bias our results [11]. When designing surveys, variables are difficult to control [5], in particular when using online questionnaires. To control variables, exclusion or randomization can be applied [17]. Exclusion means that participants who are not sufficiently experienced were excluded from the study. We ensure this by having a check question that only allows participants with some sort of practical experience to proceed with the questionnaire. Randomization means that we used a sampling technique that lead to random participants from REFSQ. Furthermore, validity is subject to ambiguous and poorly phrased questions. To mitigate this risk, we piloted the questionnaire multiple iterations to ensure that potential respondents understand our questions and intentions. Also, the complete survey protocol was reviewed by several external researchers. Another limitation is that participants not answer truthfully to the questions [17]. To address this problem, we made participation voluntary and anonymous. As participants spent personal time on answering the questionnaire we assume that those who volunteered to spend time have no reason to be dishonest [17].

External validity: External validity is concerned with the problem of generalizing the results to the target population. We assume that our results are applicable to a population that meets the sampling criteria of our survey (i.e., practitioners and researchers with experience in RE and variability). However, answers are not just influenced by the understanding of participants, but also the characteristics of companies, domains and software projects in which participants had worked. This information was recorded as part of the questionnaire. Furthermore, we only had a limited number of participants. In particular, participants at REFSQ were primarily RE researchers rather than practitioners. This means, we received a significant amount of responses from researchers and only a small amount of responses from practitioners. Therefore, we could not identify statistically significant relationships between the background of practitioners and the answers they gave.

6 Conclusions and Future Work

We presented a summary of the results obtained from a survey on RE for variability-intensive systems, conducted at REFSQ 2012. Our results indicate that even though variability is considered important, there are still many problems when dealing with it. This confirms other findings, which state that even though there has been research on change management in RE, change management is still a challenging task [18]. We also found that variability is a concern that is considered important by researchers and practitioners, rather than variability just being an academic problem. This means, special consideration should be given to variability RE practice. However, despite its significance, variability is not strictly managed. Another finding is that consistency is difficult to achieve in the presence of variability. This is probably due to the complexity induced by variability, which is supported by our finding that handling complexity is considered to be the most pressing technical issue when dealing with variability.

Furthermore, we found that a mindset change and the necessity for trained and skilled people are the most pressing non-technical issues. As already discussed in [3], those non-technical issues have also not been addressed by existing variability management approaches. Our study showed that this is also true for handling variability during RE.

The study showed that Goal-oriented RE methods are considered promising for handling variability during RE. Correspondingly, we found that goal models are considered to be a suitable modeling technique to model variability, while UML and other modeling techniques were less popular. This supports our finding that specification methods are often unsatisfactory when dealing with variability. Especially participants from industry have indicated that they are unsatisfied with existing specification methods. This has also been reported in [3].

With regards to validation and verification methods, we found that participants from academia favored requirements inspections to check variability requirements, while participants from industry considered informal desk-checks as more applicable. This is an indicator that practitioners are more in favor for less formal and ad-hoc specification methods.

The survey's findings can be a starting point for further research. For example, since we received only a small amount of responses from practitioners, the survey should be repeated to include more responses from practitioners. Also, future work to study RE for variability-intensive systems should investigate RE in real software projects. Furthermore, new approaches to support variability during RE should not focus on only one RE activity, but, if possible cover all activities.

References

1. Alves, V., Niu, N., Alves, C., and Valenca, G. 2010. Requirements Engineering for Software Product Lines: A Systematic Literature Review. *Information and Software Technology* 52, 8 (August 2010), 806-820.
2. Basili, V., Caldiera, G., and Rombach, D. 1994 The Goal Question Metric Approach. In *Encyclopedia of Software Engineering*, J. J. Marciniak, Ed. John Wiley & Sons, New York, NY, 528-532.
3. Chen, L. and Babar, M. A., "Variability Management in Software Product Lines: An Investigation of Contemporary Industrial Challenges," in *14th International Software Product Line Conference* Jeju Island, South Korea: Springer Verlag, 2010, pp. 1-15.
4. Chen, L., Babar, M. A., and Ali, N., "Variability Management in Software Product Lines: A Systematic Review," in *13th International Software Product Line Conference (SPLC)* San Francisco, CA: Carnegie Mellon University, 2009, pp. 81-90.
5. Ciolkowski, M., Laitenberger, O., Vegas, S., and Biffl, S. 2003 Practical Experiences in the Design and Conduct of Surveys in Empirical Software Engineering. In *Empirical Methods and Studies in Software Engineering*, R. Conradi and A. I. Wang, Eds. Springer Verlag, Berlin / Heidelberg, 104-128.
6. Creswell, J. W. 2002. Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. In *Proceedings of* (Thousand Oaks, CA). 246.
7. Hilliard, R., "On Representing Variation," in *Workshop on Variability in Software Product Line Architectures* Copenhagen, Denmark: ACM, 2010, pp. 312-315.
8. Jaring, M. and Bosch, J., "Representing Variability in Software Product Lines: A Case Study," in *Second Software Product Line Conference* San Diego, CA: Springer Verlag, 2002, pp. 15-36.
9. Kitchenham, B. and Pfleeger, S. L. 2002. Principles of Survey Research - Part 3: Constructing a Survey Instrument. *ACM SIGSOFT Software Engineering Notes* 27, 2 (March 2002), 20-24.
10. Kitchenham, B. and Pfleeger, S. L. 2003. Principles of Survey Research - Part 6: Data Analysis. *ACM SIGSOFT Software Engineering Notes* 28, 2 (March 2003), 24-27.
11. Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., Emam, K. E., and Rosenberg, J. 2002. Preliminary Guidelines for Empirical Research in Software Engineering. *IEEE Transactions on Software Engineering* 28, 2 (August 2002), 721-734.
12. Konrad, S. and Gall, M., "Requirements Engineering in the Development of Large-scale Systems," in *16th IEEE International Conference on Requirements Engineering* Catalunya, Spain: IEEE Computer Society, 2008, pp. 217-222.
13. Lim, S. L. and Finkelstein, A. 2011 Anticipating Change in Requirements Engineering. In *Relating Software Requirements and Architectures* Springer Verlag, Berlin / Heidelberg, 17-34.
14. Nolan, A. J., Abrahao, S., Clements, P., and Pickard, A., "Managing Requirements Uncertainty in Engine Control Systems Development," in *19th IEEE International Conference on Requirements Engineering* Trento, Italy: IEEE Computer Society, 2011, pp. 259-264.
15. Sommerville, I. and Sawyer, P. 1997. *Requirements Engineering - A good practice guide*. John Wiley & Sons, New York, NY.
16. van Gurp, J. and Bosch, J., "Proceedings of the Software Variability Management Workshop," J. van Gurp and J. Bosch, Eds. Groningen, The Netherlands, 2003, p. 142.
17. van Heesch, U. and Avgeriou, P., "Mature Architecting - A Survey about the Reasoning Process of Professional Architects," in *9th Working IEEE/IFIP Conference on Software Architecture* Boulder, CO: IEEE Computer Society, 2011, pp. 260-269.

18. van Lamsweerde, A. 2009. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. John Wiley & Sons, West Sussex, England.18. van

19. A. Vander, N. Niu, C. Alves, and G. Valença, "Requirements engineering for software product lines: A systematic literature review," *Information and Software Technology*, vol. 52, no. 8, Aug. 2010.

10 Empirical Research Fair

Online Questionnaires Programme

Tracing Requirements Interdependencies in Agile Teams

Indira Nurdiani, Samuel Fricker, and Jürgen Börstler

Blekinge Tekniska Högskola, School of Computing, SE-371 79 Karlskrona, Sweden
{indira.nurdiani,samuel.fricker,jurgen.borstler}@bth.se

1 Proposed Study

1.1 Background and Aim

The pressure of delivering a software product in timely manner and rapid requirement changes have driven many software organizations to adopt a solution that allows them to be more flexible in adapting to changes. Agile Methodology (AM) is a software development approach that tries to address the rigidity of traditional plan-driven methods. AM focuses on delivering working software on time through short and iterative development cycles. Changes to requirements are also accepted even at later stages of the development [1].

In AM, requirements are implemented in releases based on prioritization of financial value, cost, uncertainty, and risks [3]. However, practitioners find results from prioritization to be untrustworthy [5]. Requirements prioritization is further challenged by interdependencies between requirements [4]. Managing requirements interdependencies, which is an important aspect in incremental development [2], is a missing piece in AM [8].

The aim of this study is to explore the perception from agile teams regarding requirements interdependencies and uncover *in-situ* practices for handling those interdependencies. We want to study the practices that are in place from the development team point of view with ethnomethodological approaches, utilizing observations and interviews as data collection methods [6]. Through ethnomethodology we can uncover social and other aspects that can provide insights toward focused development effort improvement, as demonstrated in [7].

1.2 Expectations on Industrial Partners

We are interested in studying existing practices and techniques with respect to tracing requirements interdependencies in agile teams. The observation will be done unobtrusively during iterations at different phases of a project: close to the beginning, halfway through, and close to the end. We would also like to observe requirements related artefacts, i.e., backlogs, bulletin board, drawings on whiteboards, etc. The study also includes interviews with team members with various roles. Organization's and team members' names will be anonymized for confidentiality purposes.

From this study we want to uncover practices that contribute to requirements interdependencies management. By gaining this knowledge, we can identify issues pertaining to tracing interdependencies between requirements. Furthermore, we can uncover and retain 'good practices' that may have been overlooked in the development
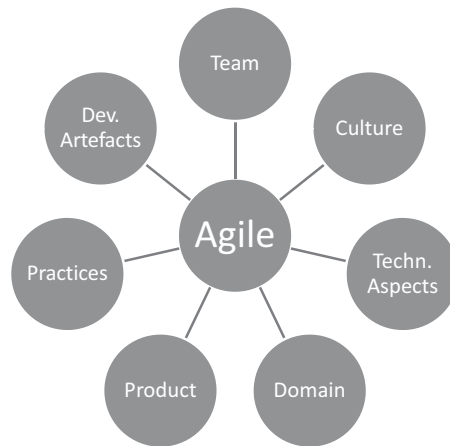
Fig. 1. Aspects to study.

team. Lastly, we want to propose an improvement initiative to support traceability of requirements interdependencies which in turn can improve requirements prioritization and release planning activities.

References

1. L. Cao and B. Ramesh. Agile requirements engineering practices: An empirical study. *IEEE Software*, 25(1):60–67, 2008.
2. P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. Natt och Dag. An industrial survey of requirements interdependencies in software product release planning. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 84 –91, 2001.
3. M. Cohn. *Agile Estimating and Planning*. Prentice Hall, Upper Saddle River, NJ, USA, 2006.
4. A. s. Dahlstedt and A. Persson. Requirements Interdependencies: State of the Art and Future Challenges. In *Engineering and Managing Software Requirements*, pages 95–116. 2005.
5. L. Lehtola and M. Kauppinen. Suitability of requirements prioritization methods for market-driven software product development. *Software Process: Improvement and Practice*, 11(1):7–19, 2006.
6. Lisa M. Given. *SAGE Encyclopedia of Qualitative Research Methods*, volume 1&2. SAGE, USA, 2008.
7. K. Rönkkö. "Yes – What Does That Mean?" Understanding Distributed Requirements Handling. In *Social Thinking - Software Practice*, pages 223–241, Cambridge, MA, United states, 2002. MIT Press.
8. G. Schalliol. Challenges for Analysts on a Large XP Project. In M. Marchesi, G. Succi, D. Wells, and W. Laurie, editors, *Extreme Programming Perspectives*. Addison Wesley Publishing Company, Boston, 2002.

What do you expect from Requirements Specifications?
An Empirical Investigation of Information Needs

Anne Gross

Fraunhofer IESE, 67663 Kaiserslautern, Germany
Anne.Gross@iese.fraunhofer.de

Introduction

Requirements specifications (RS) play a crucial role in software development projects as these documents serve as a source of information for a variety of roles involved in downstream activities like architecture, design, and testing. However, this fact poses a challenge to a requirements engineer that is responsible to create these specifications: different information needs and expectations have to be addressed that are strongly dependent on the particular role that the document stakeholders have within a project. For example, an architect requires detailed knowledge about quality and data requirements as well as technical constraints to derive appropriate architectural decisions. On the other hand, a user interface designer requires detailed information regarding characteristics of end users as well as interaction descriptions as this information has a tremendous influence on design decisions.

Today's requirements engineering approaches do not explicitly address these "role-specific" information needs. Often RS contain much more information than actually required by a certain role to perform his/her tasks. Or relevant information is hard to find and analyze in a specification as it is distributed over different sections or even different documents. Or important information is even missing. Such observations of inappropriate documentation do negatively influence an efficient and effective usage of RS as the analysis of these documents becomes time-consuming and frustrating for the document stakeholders. In the worst case, this could result in document stakeholders neglecting or ignoring the RS which finally ends in implementations of software systems that fail to meet the requirements actually documented in the RS [1].

Research Questions

In order to provide the stakeholders of RS with appropriate documents we claim that sound and empirically valid knowledge about role-specific information needs is necessary. That is, suitable studies have to be designed aiming to investigate the following research questions from the viewpoint of different development engineers:

- RQ_1: What are typical artifact types (e.g., stakeholder descriptions, quality requirements, data requirements) that should be documented in RS?

- *RQ₂*: On what level of detail should artifacts of these types be documented?
- *RQ₃*: Which notation should be used to document the artifacts?

Based on these findings we aim to investigate and analyze differences in the information needs, in particular:

- *RQ₄*: Is there a difference in the information needs between different roles such as architect, UI designer, developer, tester?
- *RQ₅*: Is there even a difference between different development engineers with the same role?

Finally, information needs might also be influenced by certain environmental factors such as expertise of the development engineers, familiarity with the project domain, internal processes, motivation, personality, etc. This is reflected in:

- *RQ₆*: What are environmental factors that influence particular information needs?

Once all these research questions have been sufficiently answered, we will be able to develop suitable solutions that enable an efficient and effective usage of RS (for example view-based requirements specifications [1] [2]).

But not only consumers of RS will benefit from these results. Also requirements engineers as authors of these documents can benefit as they can align their elicitation and specification activities in accordance with the information needs of subsequent development phases and project plan (i.e., they know WHAT and HOW to specify for each role and WHEN the role requires the information according to the project plan).

Wanted from Industry

We'd like to encourage practitioners working either as software architects, interaction / UI designer, developer, or tester in the domain of information systems to contribute to the body of knowledge about information needs.

Participation is primarily targeted to investigate research questions RQ₁ to RQ₃ and RQ₆. That is, the relevance of a given set of artifact types typically specified in the context of information systems as well as the required level of detail and suitable notation will be investigated. In addition, information about typical activities and environmental factors (domain, experience / background, etc.) will be captured. Data will be collected either in form of a (phone) interview or filling out a questionnaire. Participation will take max. 2 hours. Experiences gained during such industrial case studies will be shared among all participants of the studies anonymously.

References

1. S. Adam, N. Riegel, and A. Gross, "Focusing on the "Right" Requirements by Considering Information Needs, Priorities, and Constraints", REEW'12
2. S. Schlosser "Tool-Support for Perspective-based Views on Software Requirements Documents", Fraunhofer IESE Report 032.12/E, 2012

Applying creativity techniques to requirements elicitation: defining an enhanced EPMCreate

Luisa Mich[a], Daniel Berry[b], and Victoria Sakhnini[b]

[a]Department of Computer and Management Sciences, University of Trento, Italy
luisa.mich@unitn.it
[b]Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada
dberry@uwaterloo.ca, vsakhnini@ uwaterloo.ca

1   Introduction

Identifying high-level requirements for innovative software systems is a challenge. Real innovation needs a creative approach to be able to discover also implicit and unexpected users' requirements.

A new creativity technique is proposed, based on techniques that focus on different users' viewpoints: EPMCreate and POEPMCreate. (For more information about EPMcreate and POEPMcreate, please consult
<http://www.cin.ufpe.br/~ccte/intranet/creativity/CreativityBerry.pdf> and
<http://www.springerlink.com/content/1833346052m8351g>).

In the new technique, the analyst has to play a role similar to that of a relational therapist. Requirements engineers and representatives of the different users of a targeted system are invited to follow a multistep process in discussing a first core of high-level requirements for the system. Each step corresponds to a different relational style adopted by the chief analyst.

2   Wanted from Industry

As researchers, we are looking for an industrial project that is aiming to develop a new software system or to innovate an existing one to gain competitive advantage. The company should provide a seed set of requirements, constraints, and desiderata that will be used to start the relational requirement session. The goal of this first application of the new technique is to test its usability and design in a real context (See the next page for a copy of the poster presented at the Empirical Fair at REFSQ'2012).

The domain of the software system is not relevant, while a real commitment to work on the initial requirements is a must. For the first application, two requirements engineers or one engineer and a representative of the marketing department of the company would be the best choice. Using only a single subject or more than two could be discussed with the researchers.

A couple of two-hour sessions are needed for this first test. The expected outcome for the company would be a number of innovative requirements.

Applying Creativity Techniques
to Requirements Elicitation:
Defining an Enhanced EPMcreate

Luisa Mich, University of Trento, Italy
Daniel M. Berry and Victoria Sakhnini, University of Waterloo, Canada

You are trying to develop your next **innovative software system**.

Finding **innovative requirement ideas** for your software system requires **creativity**.

How do you encourage your analysts to be more creative,
to think more out of the box for the innovative software system?

Brainstorming for requirement ideas is a popular and effective method for encouraging analysts to be more creative in generating requirement ideas for your software system.

Perhaps you have felt that there are better ways to encouraging creativity in generating innovative requirement ideas. **You are right!**

We have demonstrated in a number of **experiments** that **EPMcreate** is more effective than brainstorming to generate requirement ideas.

EPMcreate works **better than brainstorming** because EPMcreate gets the analysts to **systematically explore the space** of requirement ideas while brainstorming leaves analysts wandering aimlessly in the same space.

EPMcreate leads its users to focus on all combinations of the viewpoints of software system's **stakeholders**.

We have also demonstrated that two optimizations of EPMcreate, one involving **fewer steps** and one involving **smaller groups**, are even more effective in helping a group of analysts to generate innovative requirement ideas.

These optimizations are **cheaper** to deploy than is EPMcreate, because they require less time or fewer people.

We are considering other optimizations and seek industrial organizations in which to empirically test their effectiveness in enhancing requirements elicitation.

The benefits to you for allowing us to do the tests in your organization are that:

- Your **people will learn EPMcreate** and the two optimizations that are immediately usable.
- Your people may end up learning an even better optimization.

Creativity is allowing yourself to make mistakes.
Art is knowing which ones to keep.
Dilbert

If you are interested in participating, please **contact**:
Luisa Mich at <luisa.mich@economia.unitn.it> in Europe,
Daniel Berry at <dberry@uwaterloo.ca> in North America, or either anywhere

Supporting Client-Developer Feedback Loops in Agile Requirements Engineering by means of a Mobile RE Tool

Maya Daneva[1], Nelly Condori-Fernandez[2], Norbert Seyff[3]

[1,2]University of Twente, The Netherlands [3]University of Zuerich, Switzerland

[1]m.daneva@utwente.nl, [2]n.condorifernandez@utwente.nl, [3]seyff@ifi.uzh.ch

Abstract. This poster paper presents an exploratory study on the use of the iRequire tool in support of client-developer feedback loops in agile projects. The iRequire tool is a smart phone application that has already been evaluated in three pilot studies. In this study, we will use iRequire to analyze the types of requirements blogging behavior in an agile RE process for the specific purpose of understanding helpful and unhelpful practices in agile requirements elicitation. Helpful are those that enhance business value generation, while unhelpful are the ones that hamper it.

Keywords: Requirements engineering, stakeholders collaboration, users' feedback, requirements elicitation, micro-blogging

1 Introduction: What is our research about and why it is important?

Clients' involvement by means of quick feedback loops is recognized to be both the strongest and the weakest point in agile RE processes. The strength lies in the increased business value generation through the mechanisms for frequent question-answering sessions that agile processes presuppose. However, prior research pointed out that value generation is impeded due to difficulties in getting clients consistently involved when developers need them. Specifically small and medium size client companies often face time pressure and work burdens because while serving on-site on a project, they leave their regular office work undone. As a result, many agile developer teams find it problematic to enlist their client's collaboration, which, in turn, impacts the flow of business value throughout the agile project. To overcome this issue, companies put either an "on-site developer" in the client organization, or establish some phone-based communication practices to remain in touch with their clients on everyday basis. However, these approaches are only partial solutions from the client's perspective. In this study, we therefore focus on an alternative approach which leverages the fact that many clients use smart phones in their day-to-day business communication and the use of their phones can be extended to serve agile RE purposes. Specifically, we focus on the tasks of collecting user feedback on software product functionality that is delivered in each iteration of an agile project. We suggest a mobile RE tools be used to enable clients to blog their requirements in a

location-and-time-independent manner. The requirements can (i) be in a variety of formats, e.g. voice, text, photographs or videos, and (ii) pertain to new applications or to already existing systems that are subjected to follow-up improvements. The goal of this study is to get a deeper understanding of what helpful and unhelpful behaviors on client's and on developer's sides the use of such a tool will promote in agile project. To the best of our knowledge, mobile RE tools have not yet been deployed for supporting agile RE. We propose to carry out an exploratory case study in which we use a real-life project with intensive client-developer interactions, and apply one specific mobile tool, iRequire, that has already been evaluated in three small-scale pilots. We will analyze the types of requirements blogging behavior in an agile RE process in the organization for the purpose of understanding helpful and unhelpful practices in agile requirements elicitation. Helpful are those that enhance business value generation, while unhelpful are the ones that hamper it.

2   Wanted from industry

We seek collaboration with any agile organization willing to adopt the iRequire tool for at least three iterations of their agile project. We would need at least (i) two clients or marketing managers who represent the client organization, and (ii) one member from the agile development team, be it a project manager or a developer. The iRequire tool would be deployed in support of requirements elicitation and collection tasks that occur between iterations. We estimate a max of 8 person/hours distributed over 6 weeks of time. This time will be spent on letting the clients use iRequire, and interviewing them, to understand the kinds of requirements that were elicited and the helpful and unhelpful behaviors from the client's and developer's site in the process. We consider our research to be of value to the company, because it may shed light into those client-developers interactions that are critical for business value generation, so that company might search for ways to enhance their role in the agile process.

2   Work Plan

Our work plan includes the following: First we will train the practitioners (i.e. clients and developers) on the iRequire tool (which is a mobile application on a smart phone). Second, once practitioners use it, we will review the requirements collected and classify them according to types of problems they address. We will also interview the clients and the developers about what they found as helpful and unhelpful behaviors that they used in the process of blogging requirements. The interview data will be analyzed by means of the Grounded Theory approach.

Using E-mails and Phone Calls to Resolve Requirements Engineering Issues: Which Works Best and for Which Type of Issue?

Maya Daneva

University of Twente, Drienerlolaan 5,
7522 AE Enschede, The Netherlands
m.daneva@utwente.nl

Abstract. This poster treats the requirements engineering process as a series of conversations and acknowledges that many of these conversations happen via email and over the phone. However, some of these conversations are critical, for example those about requirements issue resolution. The goal of this proposal is to understand what renders email unproductive and for what kind of issues would phone calls be a better alternative. Also, how to recognize the time when it is better to stop emailing and consider face-to-face meetings or phone calls.

Keywords: Requirements engineering, communication, collaboration, requirements negotiation, stakeholders interaction.

1 Introduction

Requirements Engineering (RE) relies on conversations [1] to occur. While RE text books implicitly assume that these conversations are face-to-face, in most projects, a major portion of the requirements conversations and issue resolution take place over email and over the phone (even when the involved parties work on the same floor in an office). However, recent studies in psychology and in organizational behavior that investigated projects stakeholders' email use for the purpose of issue resolution, indicate that email is a convenient mechanism for issue avoidance. The 2009 study from the University of Massachusetts Amherst found that this is especially true in cases in which stakeholders face decision fatigue [2] (too many decisions are addressed to them to be processed in too few hours) or work pressure, and also that email is conductive to stakeholder's use of deception.

In requirements conversations, it is often easier, faster, less stressful, and less confrontations to have critical or challenging email versus a live one-on-one with a counterpart. As a result, many RE staff members experience unproductive strings of back-and-fort emails of texts that could have been stopped in round two, but continued. Example of known problems are the tendency to prolonged debate, the promotion of reactive responses and the easiness of misreading tone and context. This email-based mechanism in conversations seems therefore detrimental to critical RE processes that are likely to be focused on quick issue resolution, most notably,

requirements negotiation (e.g. requirements prioritization, requirements change evaluation). In this study, we set the goal to understand the critical points in time at which RE stakeholders abandon unproductive strings of emails and pick up the phone, or better – call for a personal meeting. The overall expected outcome will be to distill some heuristics for helping choose the email over the phone or vice versa when in need to resolve requirements issues.

2 Wanted from industry

We welcome the collaboration with an organization willing to provide access to at least 10 practitioners involved in different roles in the RE process. The practitioners should (1) have at least three experiences in unproductive emails, and (2) have shown courage and picked up the phone for the sake of resolving things more efficiently. We will administer an open-ended questionnaire to understand (i) some important attributes of the requirements issues the practitioners had to resolve, (ii) the inconveniences the email-strings had created, and (iii) the reasoning that the practitioners used to make the judgment that a phone call or a meeting would be more productive. We plan to have a 30 min interview with each practitioner. This amounts to 5 person/hours that would take place over a period of a month. We consider our research to be of value to the company, because it might generate ideas on how to improve the balance between email and phone use in conversation aiming at requirements issue resolution.

2 Work Plan

Our work plan includes the following: First, each practitioner will be introduced into the goals of the study and will be asked to answer no more than 10 questions. The interviews will be transcribed and analyzed. We will use qualitative analysis techniques (e.g. grounded theory) for interview data processing. The resulting heuristics will be evaluated for external validity in follow-up studies.

References

1. Goleman, D. Social Intelligence: the New Science of Human Relationships, Betam (2006)
2. Baumeister, R. F., Tierney, J. Willpower: Rediscovering the Greatest Human Strength, Pinguin Press (2011)

Patterns of Requirements related Communication

Eric Knauss, Daniela Damian

University of Victoria, Canada
{erickn, danielad}@cs.uvic.ca

Introduction. Effective collaboration during Requirements Engineering is essential for project success and yet very difficult. This collaboration includes the discussion and negotiation of requirements with different stakeholders, as well as deriving, assigning, and scheduling tasks and subtasks from these requirements. Although existing requirements management tools offer some support to this collaboration, practitioners rely on a combination of collaboration tools such as email and issue-trackers. Often, this leads to missing a complete view on the state of the requirements-related discussion as well as a lost opportunity in leveraging the wealth of requirements-related communication data available in projects.

Collaboration with the Industry Partner. In our research we aim on investigating patterns of requirements related discussions that can help projects continually monitor the health of requirements-driven collaboration. Based on a framework for analyzing requirements-driven collaboration [1], we offer to analyze requirements-driven communication by developing stakeholder requirements centric social networks (RCSN). Industry partners could benefit by pursuing a number of analyses for:

- *Broker identification:* Identifying the brokers of requirements related information to make project managers aware of critical people in a project.
- *Expertise seeking:* Analyzing the communication and assignments to tasks related to requirements to help find experts for a given topic.
- *Diagnosing coordination:* Analyzing the alignment and correlation between different social networks provides valuable information to managers about:
  - *Socio-technical congruence,* as an example of a measure that can identify gaps in coordination. It allows managers to better align the social structure of an organization with the technical dependencies among requirements.
  - *The health of requirements and their development* – analyzing requirements related-discussion allows identifying problematic requirements where the coordination is burdened by ongoing efforts to clarify the requirement.

Our research aims at identifying patterns of requirements based communication in software projects and will extend the analysis framework with automatic classification of discussion items (analogous to identifying security issues in [2]).

1. Damian, D., Kwan, I., Marczak, S.: Requirements-driven collaboration: Leveraging the invisible relationships between requirements and people. In: Mistrík, I., Grundy, J., Hoek, A., and Whitehead, J. (eds.) Collaborative Software Engineering. 57-76. Springer, Berlin Heidelberg (2010).
2. Knauss, E., Houmb, S., Schneider, K., Islam, S., Jürjens, J.: Supporting Requirements Engineers in Recognising Security Issues. In: Proceedings of REFSQ'11. Springer, Essen, Germany (2011).

Requirements Elicitation Driven by End-Users

Alessia Knauss, Daniela Damian

University of Victoria, Canada
{alessiak, danielad}@cs.uvic.ca

Introduction. Requirements engineering, especially for existing software systems, relies on effective capturing of stakeholder needs. End-users of the system are a valuable source of creativity not sufficiently taken into account in conventional requirements engineering methods. We study requirements elicitation driven by end-users: End-users are not considered to be passive sources of requirements, but as active participants of requirements elicitation. By allowing end-users to articulate their needs and to propose suggestions for improvement the chances of enhancing user acceptance of the system are increased. Existing approaches based on social media show promising support for end-user driven requirements elicitation.

Research Goal. Our goal is to investigate which factors have an impact on the quality of end-user driven requirements elicitation. For this, we want to analyze requirements-driven collaboration [1], i.e. how end-users discuss requirements in a group. Among other factors, we want to investigate in this context how the following factors impact the effectiveness of end-user driven requirements elicitation:

- *Multimedia content* – we assume that context rich multimedia representation of requirements improves the end-users' ability to identify tacit needs and is an efficient alternative to textual requirements documentation [2].
- *Seeding* – we assume that end-users will find it easier to add new requirements, if there already exist requirements that were seeded into the social media.

Collaboration with the Industry Partner. We are looking for an industry partner who is ideally constantly improving an existing, large, feature-rich software system, used by many different user types. Together, we want to explore how these new media allow integrating end-users in the requirements engineering process in a case study. With the help of the evaluation results we hope to improve the industry partner's requirements engineering process. Based on the increased involvement of end-users we are confident that this will increase chances to build software that satisfies users.

1. Damian, D., Kwan, I., Marczak, S.: Requirements-driven collaboration: Leveraging the invisible relationships between requirements and people. In: Mistrík, I., Grundy, J., Hoek, A., and White-head, J. (eds.) Collaborative Software Engineering. pp. 57-76. Springer-Verlag, Berlin Heidelberg (2010).
2. Brill, O., Schneider, K., Knauss, E.: Videos vs. Use Cases: Can Videos Capture More Requirements Under Time Pressure? In: Wieringa, R. and Persson, A. (eds.) Proceedings of the 16th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ '10). pp. 30-44. Springer, Essen, Germany (2010).

Part III

REFSQ 2012 Doctoral Symposium Proceedings

11 Preface

Editor

Barbara Paech
University of Heidelberg, Germany, paech@informatik.uni-heidelberg.de

Report on the
Second REFSQ Doctoral Symposium

Barbara Paech

University of Heidelberg, Germany
paech@informatik.uni-heidelberg.de

In 2012 REFSQ has hosted the second Doctoral Symposium for Ph.D. students working in the general area of Requirements Engineering. The symposium was organized by the author, and intended to bring together Ph.D. students with the double purpose of encouraging networking and the establishment of links at an early stage in their careers, and of providing valuable advice from a panel of senior researchers on how to best bring the Ph.D. work to fruitful (and timely!) completion.

In response to the Call for Papers, ten submissions were received from as many Ph.D. students, each consisting of two elements:

1. a research abstract describing the problem addressed, its relevance for research and practice, an outline of the intended solution, some consideration on the novelty of the proposed solution, and an outline of the research method applied and of the current stage of the work;
2. a recommendation letter by one of the supervisors, establishing relevance for Requirements Engineering research and presenting a general overview on the student's progress.

All proposals were independently reviewed by two members of the Symposium's Program Committee, who provided recommendation for acceptance or rejection. As a result of this process, seven submissions were accepted for presentation.

In line with the REFSQ tradition, and with the overall goal, the Symposium was planned for special emphasis on discussions and interaction, rather than on presentations. To further encourage discussion not only with panelists and other participants to the Symposium itself, but also with the general audience of REFSQ, authors were invited to prepare posters to be displayed in the common area during the main REFSQ conference (so that participants to both the scientific and the industrial track of the main conference could examine them).

On the 19th of March, 2012, the Doctoral Symposium took place. Attendants included the seven presenters, a group of senior researchers serving as advisors, and some students who had not submitted a proposal, but who were attending the Symposium in preparation for future developments of their work. Each presenter was allocated 50 minutes, with 20 minutes for presentation and 30 for discussion. As is typical of the REFSQ spirit, the discussions were very lively, both by the panelists and by fellow students. Minutes of the various discussions were recorded during the symposium by the participants, and provided to the presenters; this ensured that the students could benefit from a complete and clean trace of the whole discussion.

In addition to the presentations, the Symposium hosted a micro-tutorial offered by Dan Berry about how to complete a Ph.D. on time — reinforcing the main message of focusing on a well-defined problem before venturing into extending a proposed solution to loosely related issues. After the micro-tutorial, a brief session was devoted to collecting feedback and impressions from the students and panelists, with the purpose of improving the process for next year's edition.

After the Symposium, the seven presenters were invited to submit a revised version of their research abstracts, taking into account the advice and suggestions received, as well as any further progress in their work that might have happened since the original submission. The reader will find these extended abstracts (besides one) in the following pages.

We would like to thank the members of the REFSQ Doctoral Symposium Program Committee, who helped in the selection process and provided initial feedback to the students, and the panelists who participated in the lively discussions.

Last, but not least, we gratefully acknowledge the excellent logistic support we received from the local organizers, and notably the assistance of Vanessa Stricker.

Doctoral Symposium Organization

Program Committee
Dan Berry, University of Waterloo, Canada
Sjaak Brinkkemper, University of Utrecht, Netherland
Vincenzo Gervasi, University of Pisa, Italy (co-chair)
Tony Gorschek, Blekinge Institute of Technology, Sweden
Marjo Kauppinen, Helsinki University of Technology, Finland
Camille Salinesi, University Paris 1 Panthéon – Sorbonne, France

Panelists
Dan Berry, University of Waterloo, Canada
Sjaak Brinkkemper, University of Utrecht, Netherland
Samuel Fricker, Blekinge Institute Technology, Sweden
Marjo Kauppinen, Helsinki University of Technology, Finland
Barbara Paech, University of Heidelberg, Germany
Camille Salinesi, University Paris 1 Panthéon – Sorbonne, France
Kurt Schneider, University of Hannover, Germany

Local organization & proceedings
Vanessa Stricker, paluno, Germany
Wilhelm Springer, University of Heidelberg, Germany

Presenters
Noorihan Abdul Rahman, Universiti Teknologi Malaysia, Malaysia

Elizabeth Bjarnason, Lund University, Sweden
Alexander Delater, University of Heidelberg, Germany
Pariya Kashfi, Chalmers University of Technology, Sweden
Marko Komssi, Aalto University School of Science and Technology, Finland
Cyril Mauger, Public Research Centre Henri Tudor, France
Cristina Ribeiro, University of Waterloo, Canada

12 Doctoral Symposium

Doctoral Symposium Programme

Requirements Elicitation Technique for Social Presence in Collaborative Activities in Support of E-learning Domain

Noorihan Abdul Rahman[1,1], Shamsul Sahibuddin[2],

[1] Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA Malaysia, Shah Alam, 40450 Shah Alam, Malaysia
[2]Advanced Informatics School, Universiti Teknologi Malaysia International Campus, Jalan Semarak, 54100 Kuala Lumpur, Malaysia
noorihan@kelantan.uitm.edu.my, shamsul@utm.my

Abstract. Requirement Engineering (RE) is an initial stage in software production and essential for achieving software quality. Proper identification of requirements technique is crucial to ensure software requirements are translated accurately by the stakeholders. Requirements elicitation enables requirements engineers and developers to successfully capture the right criteria according to users' interests as well as minimizing requirement error for development process. This paper addresses a new requirements elicitation technique to support requirement from human dimension in collaborative activities. Social presence is taken as the research element in collaborative activities. This is to visualize how the proposed technique may assist in capturing problem-specific domain such as social presence. E-learning is chosen as the case study to foresee how requirements engineering may enhance its knowledge by identifying processes to extract social presence as requirements features from user's perception. This paper also outlines progress and future work that will be carried out by the researcher.

Keywords: Requirements elicitation technique, requirements engineering, social presence, social interaction, collaborative activities, documents.

1   Problem Statement

In producing software, Requirements Engineering (RE) plays its part in eliciting requirements, refining requirements, prioritizing requirements and establishing requirements. RE consists of a few processes like requirements elicitation or requirements discovery, requirements analysis and reconciliation, requirements representation or requirements modeling, requirements verification and validation and requirements management [1]. RE is considered as an important process in software

---

development since identification of right requirements can aid requirements engineers and developers to achieve desired software specification by minimizing requirements errors. Therefore, prevention of requirement error must be detected earlier to prevent from potential and critical risks in the future [2]. A proper requirements elicitation technique may contribute to accurate requirements before implementation starts. There are existing requirements elicitation techniques available to maintain requirements' consistency, accuracy as well as ambiguity [3-5], however existing methods are focusing on general problem domain and there is an opportunity to improve elicitation technique to support human activities in order to address problem-specific domain [6]. From this background study, we believe that a new requirements elicitation technique is possible to be introduced to address problem-specific domain in order to support human activities and a sub research question for this issue is, 'Is it possible to introduce a new requirements elicitation technique to address the domain that is related to human activities?'

In collaborative software such as E-learning, for example, requirements engineer and related stakeholders need to understand what are the requirements required by users [7-9]. Getting requirement from human activities is considered as a challenge since it is generated from human's expression and feedback. Requirements elicitation technique can assist stakeholders to express their ideas on how to obtain software features. In E-learning, for instance, developer needs to understand the idea of cognitive science in order to permit the understanding of social interaction value in E-learning communities. Collaborative activities in E-learning allow learners to communicate for knowledge sharing during learning process regardless of ways of interaction in E-learning platform [10]. Learners can perform activities like discussion, online quiz and assignment, announcement in online forum, chatting with friends and other related tasks which can help them to share their knowledge and ideas [11-14].

However, there is a challenge in sustaining learning interest in E-learning. Some students lack of motivation to interact and feel insecure to express their opinions in E-learning [8, 15, 16]. There is also an issue in preserving the usability of E-learning among students and hence reduce the flexibility to learn anytime anywhere. Having said that, we believe that there is an opportunity in enhancing requirements elicitation technique for collaborative application especially for E-learning domain. This is also to address the issue of technique in elicitation for only general domain. Improvement of proposed elicitation technique in this research is hoped to result a technique that cater problem-specific domain issue such as E-learning domain. Another sub research question is 'Is the new requirements elicitation technique able to improve social presence in collaborative activities in E-learning domain?'

2 Relevance or Motivation

Requirements elicitation is a very important phase in RE. Having malfunction of requirements elicitation may invite project failure in software development [17]. There are several reasons why this topic has been initiated as a research topic. First and foremost, the issue of problem-specific domain [6] such as social presence in E-

learning application has triggered the issue of weakness in existing requirements elicitation techniques. It gives the opportunity to explore the knowledge of requirements elicitation technique as well as identifying steps needed in order to capture social presence as requirements in E-learning. Social presence in an E-learning can be expressed by a feeling of being there or being with others as if in face to face classroom. It can be defined as the sense of "being there" [18] or psychologically present [19] with others.

The development of social presence involved with demonstrative, dynamic and cumulative [20]. Demonstrative involves the action from the person such as posting messages or any online activities that may leave his remark online. Whereas, frequency of the interaction, number of time spent for interaction as well as interests on the interaction may lead to how dynamic is the social presence in the online application. Social presence also can increase overtime whereby one person may get familiar with another person through their history of discussion or previous communication in online application. The person who is not familiar with another person will have less interaction among them since their social presence has not been developed previously in the online environment.

As for the second motivation, this research has enlightened the importance of eliciting human activity in E-learning since E-learning portrays social interaction through online activities [21]. By investigating element of social presence in online activities, requirements elicitation techniques should provide the way for stakeholders to mutually understand requirements for social presence and hence maintain connectedness among users in E-learning.

Thirdly, a proper requirements elicitation technique may mitigate stakeholders in obtaining more accurate requirements [4, 6, 22-24] of social presence in E-learning. Therefore, there is a reason why there is a need to prepare a complete set of requirements specification in requirements document in order to understand users' request which is related to human activities in E-learning.

Fourthly, there is a prospect to increase the quality of elicitation technique for social presence value in E-learning since existing elicitation techniques are more appropriate for capturing requirements with these criteria; consistency, correctness and ambiguity. It is a challenge for requirements engineer to carry out a technique that can help them to elicit human-related requirement such as social presence. Requirements engineer may find some difficulties in getting requirement related to human experience and feeling. Users may find it easier to articulate their requirements about technical requirement rather than human activity with the software.

3 Proposed Solution

This research proposes requirements elicitation technique for supporting social presence in collaborative application. E-learning has been selected as the domain to identify and discuss collaborative activities involved in encouraging social presence. The proposed technique will involve some processes to ease requirements engineers and related stakeholders to capture requirements in social presence. The proposed technique will also permit users to address social interaction requirements voluntarily.

The solution is going to complement the research question of this study, which is to answer, 'Is the new requirements elicitation technique able to capture social presence as a requirement in collaborative application?' By focusing on how to implement a new technique, we believe that sub research questions in Section 1 can also be achieved in this study.

For the elicitation process, the researchers are planning to interpret the meaning of social presence requirements into a format which is understandable by the requirements engineers and related developers. The research members are doing progress in determining the sufficiency of whether to apply ontology or to use a simple translation tool which can help requirements engineers and developers to produce a requirements document for social presence. Ontology will be used in elicitation process in order to explicitly described terms and concepts of social presence. From the preliminary result of the selected E-learning, social presence terms and concepts will be interpreted using description of ontology. XML will be used to help the translation of social presence element to a set of requirements in E-learning domain and hence a category of social presence requirements will be introduced into existing requirements document. The summary of proposed solution can be illustrated in Fig.1.
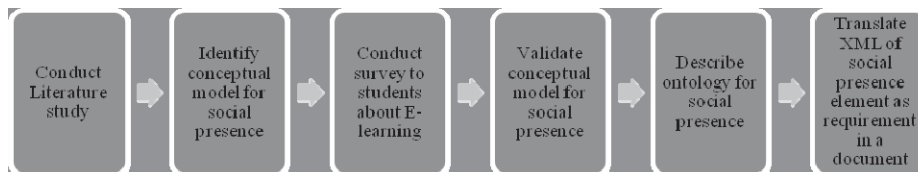
Fig. 1. Proposed Solution for Requirements Elicitation Technique.

In order to validate the conceptual model, three E-learning applications will be used as domains to test the new requirements elicitation technique for extracting social presence in collaborative application. Those three different E-learning systems will be taken from three different universities in Malaysia. The model will therefore be generalized according to the three E-learning systems.

4 Novelty

The contributions of this research are as follows:
i.   Contribute to suitable requirements elicitation technique in requirements engineering field.
ii.   Include social presence category in requirements document in order to improve software requirements specification.
iii.   Improve social requirement extraction technique in elicitation activity.
iv.   Improve students' anticipation in E-learning usage by imposing social presence features in E-learning design.

5 Research Method

Currently, quantitative study has been carried out to produce a preliminary result. For this research, 5 variables have been used for designing the conceptual framework and its numerical data is organized and analyzed using statistical tools such as SPSS and AMOS. Quantitative can be used to measure the relations [25] between 5 variables that has been identified at recent stage using literature study.

For the preliminary result, the questionnaires were given to 130 respondents and the sample is resulted from a stratified sampling from the Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA Kelantan. AMOS is used to analyze data that has been pulled from SPSS. AMOS is useful in order to carry out Confirmatory Factor Analysis (CFA) for conceptual model which is visualized in Fig. 2. Numerical analysis is done quantitatively in order to identify whether 4 independent variables mentioned have relations with the dependent variable. The questionnaires have been modified from Tao [26] and Gunawardena [27]. Tao has conducted a study on a relationship between motivation and social presence in online class while Gunawardena did a study on the relationship between satisfaction and social presence in computer-mediated conferencing environment. Both study concluded the existence of social presence value in online communication and face to face class respectively.
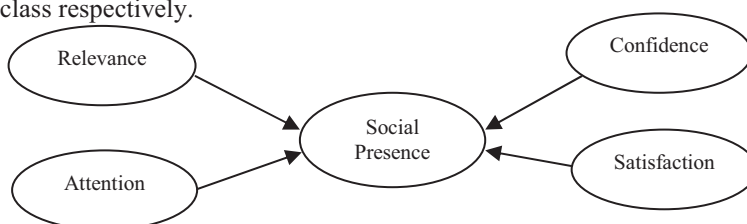
Fig. 2. Conceptual Framework of Social Presence in E-learning.

The questionnaires has been designed based on ARCS Model of Motivational Design introduced by Keller [28]. Keller has identified four steps to encourage and sustain motivation in the learning process. The model which is known as ARCS model comprised of Attention, Relevance, Confidence and Satisfaction. In conducting a literature study on social presence and ARCS Model, the authors have identified the elements of social presence in the model. Therefore, the questionnaire adopted from Gunawardena and Tao is possible for this research as well as identifying attention, relevance, confidence and satisfaction as possible factors in social presence.

The sample size has been determined based on Hair et al. [29] which stated that five or fewer constructs, each with more than 3 items per construct and high item communalities can be adequately estimated with sample size as small as 100 to 150. The total of 130 respondents has answered the closed-ended questionnaire based on the element of social presence and the questionnaires have been categorized according to ARCS categories. The questionnaires have 5 sections. Section 1 is designed to capture students' feedback on attention in using i-Learn for their study. Section 2 is design to measure students' response on relevancy of i-Learn content in terms of subjects, assignments, activities and tasks given for i-Learn activities. Section 3 follows with seeking students' confidence level in using i-Learn. Section 4 is design

to obtain satisfaction among students using i-Learn. Section 5 is to capture social presence level among students in i-Learn environment.

Table 1. Latent Constructs and Measuring Items.

Variable	Latent Construct	Number of Measuring Items
Independent	Attention	9
Independent	Relevance	10
Independent	Confidence	18
Independent	Satisfaction	4
Dependent	Social Presence	13

The variables involved in the study consist of independent variables and dependent variables. Independent variable which is also known as exogenous variable comprised of 'Attention', 'Relevance', 'Confidence' and 'Satisfaction'. Dependent variable or endogenous variable in this study is 'Social Presence'. From these variables, measurement models have been identified by identifying latent constructs and respective measuring items. Table 1 lists down number of measuring items for each of the latent constructs that are analyzed during measurement model. Further processes for research design are still in discussion phase and ongoing in this research since the relevancy of having those processes have not been finalized and justified by research members.

6   Progress and Future Work

This research has carried out a preliminary study for identifying social presence element in selected E-learning domain, which has been elaborated previously in Section 5. There will be a follow up of identifying social presence element from experts. From there, the researchers will translate social presence elements into meaningful statement and this information is added into software requirements specification. Three different E-learning from three different universities will be used as the domain to see whether conceptual model is accepted for finding the relations of potential factors of social presence in the study.

The study is still at the initial stage of identifying potential factors for social presence. Researchers need to re-address the issue of designing requirements elicitation technique since the technique is expected to generate social presence requirements from E-learning users. In doing so, researchers are now refining a suitable process to achieve the objective of obtaining the requirements for social presence. It is important for the researchers to compare a new requirements elicitation technique with existing requirement elicitation techniques available in the literature. By having the comparison of the elicitation techniques, this study is looking forward for social presence requirement as the result of the elicitation technique in supporting E-learning application and thus, may differentiate the technique with the existing ones. Thus, there will be a contribution of knowledge for the area of RE for elicitation process specifically for E-learning domain.

References

1. Laplante, P.A.: Requirements Engineering for Software and Systems,ed. Auerbach Publications (2009)
2. Zowghi, D.: Does global software development need a different requirements engineering process. Citeseer (2002)
3. España, S., et al.: An empirical comparative evaluation of requirements engineering methods. Journal of the Brazilian Computer Society. 16(1): p. 3-19 (2010)
4. Kitamura, M., et al.: A Supporting Tool for Requirements Elicitation Using a Domain Ontology. Software and Data Technologies. p. 128-140 (2009)
5. Liu, C.L.: Ontology-based requirements conflicts analysis in activity diagrams. Computational Science and Its Applications ICCSA 2009. p. 1-12 (2009)
6. Kaiya, H., Saeki M.: Using Domain Ontology as Domain Knowledge for Requirements Elicitation. In: 14th IEEE International Requirements Engineering Conference(RE'06). IEEE Computer Society (2006)
7. Yeo, R.: Problem-based learning: lessons for administrators, educators and learners. International Journal of Educational Management. 19 (7): p. 541-551 (2005)
8. Taradi, S.a.K., et al.: Blending problem-based learning with Web technology positively impacts student learning outcomes in acid-base physiology. 29: p. 35–39 (2005)
9. Kinshuk, T.L.: Application of Learning Styles Adaptivity in Mobile Learning Environments. In: Third Pan-Commonwealth Forum on Open Learning. Dunedin, New Zealand (2004)
10. Fadel, L.M., Dyson, M.C.: Enhancing interactivity in an online learning environment. In: Lecture Notes in Computer Science. 4663: p. 332 (2007)
11. Graf, S. Kinshuk: Analysing the Behaviour of Students in Learning Management Systems with Respect to Learning Styles. 93: p. 53–73 (2008)
12. Helic, D., Krottmaier, H., Scerbakov, N.: Enabling Project-Based Learning in WBT Systems. International Journal on E-Learning (IJEL) 2005. 4(4): p. 445-461 (2005)
13. Kreijns, K., Kirschner, P.A., Jochems, W.: Identifying the pitfalls for social interaction in computer-supported collaborative learning environments: a review of the research. Computers in Human Behavior. 19(3): p. 335-353 (2003)
14. Liu, Y., Lin, F., Wang, X.: Education practice and analysing of students in a Web-based learning environment: an exploratory study from China. Online Information Review. 27(2): p. 110-119 (2003)
15. Aziz, A.A., et al.: A Malaysian Outcome-based Engineering Education Model. International Journal of Engineering and Technology. 2(1): p. 14-21 (2005)
16. Yusof, K.M., et al.: Problem Based Learning in Engineering Education: A Viable Alternative for Shaping Graduates for the 21st Century? In: Conference on Engineering Education. Kuala Lumpur (2004)
17. Kausar, S., et al.: Guidelines for the selection of elicitation techniques. IEEE (2010)
18. Heeter, C.: Being there: The subjective experience of presence. Presence: Teleoperators and virtual environments.1(2): p. 262-271 (1992)
19. Gefen, D., Karahanna, E., Straub, D.W.: Trust and TAM in online shopping: An integrated model. Mis Quarterly. p. 51-90 (2003)
20. Kehrwald, B.: Being online: Social presence as subjectivity in online learning. London Review of Education. 8(1): p. 39-50 (2010)
21. Uden, L.: Activity theory for designing mobile learning. Int. J. Mobile Learning and Organisation (2007)
22. Dzung, D.V., Ohnishi, A.: Ontology-Based Reasoning in Requirements Elicitation. IEEE. (2009)
23. Farfeleder, S., et al.: Ontology-driven guidance for requirements elicitation. The Semantic Web: Research and Applications. p. 212-226 (2011)

24. Shibaoka, M., Kaiya, H., Saeki, M.: Goore: Goal-oriented and ontology driven requirements elicitation method. Advances in Conceptual Modeling Foundations and Applications. p. 225-234 (2007)

25. Teruel, M.A., et al.: A Comparative of Goal-Oriented Approaches to Modelling Requirements for Collaborative Systems (2011)

26. Tao, Y., The relationship between motivation and online social presence in an online class. University of Central Florida Orlando, Florida (2009)

27. Gunawardena, C.N., Zittle, F.J.: Social presence as a predictor of satisfaction within a computer-mediated conferencing environment. American Journal of Distance Education. 11(3): p. 8-26 (1997)

28. Keller, J.: Development and use of the ARCS model of instructional design. Journal of Instructional Development. 10(3): p. 2-10 (1987)

29. Hair, J.F., Anderson, R.E., Tatham, R.L., Black, W.C.: Multivariate Data Analysis, 7thEdition. Pearson Prentice Hall (2010)

Integrating Requirements Engineering with Software Development - A Research Abstract

Elizabeth Bjarnason

Department of Computer Science, Lund University, Lund, Sweden
elizabeth.bjarnason@cs.lth.se

Abstract. Software development companies operating in market-driven domains need to deliver new and appealing software products at an increasing rate in order to stay competitive. This requires fast and efficient development of software for which the requirements are based on ever-changing market demands. Agile development claims to achieve increased development efficiency by performing the requirements engineering (RE) activities concurrently with design, planning and testing in an integrated fashion. However, coordination and communication is often reported as a challenge both for agile and for traditional RE practices. Increased insight into the factors affected by integrating RE may allow tailoring the degree of RE integration to suit specific project context, e.g. size, rate of requirements change, domain etc., and thereby support increased efficiency in software development. The aim of this research is to develop methods for assessing the level of RE integration and techniques for improving the integration of requirements. The research is performed in collaboration with industry and the developed methods and techniques will be empirically evaluated in an industrial setting.

Keywords: Integrated requirements engineering, software development, agile development, process assessment, alignment, testing

1 Introduction

Companies operating in market-driven domains face the challenge of balancing high requirements volatility [10] and uncertain cost estimates [10] with releasing software within a critical market window [18], thus, making time to market an important competitive factor. In addition, in large software development companies, communication and knowledge share [4] between organizational units and roles [11] is vital for enabling efficient development of competitive software. Requirements engineering (RE) can aid the software development life cycle [5] both by enabling decision making on which requirements to implement and by supporting clear communication of these requirements to the relevant development roles. However, this requires the RE activities [5] and roles [11] to be well coordinated with the rest of software development, e.g. design, implementation and testing. Requirements that are not aligned with design and with the amount of available resources are likely to cause problems and lead to delays, wasted effort and issues with software quality [1, 5]. In

addition, failure to communicate requirements and changes to them to all relevant parties, e.g. developers and testers, can lead to similar problems [3].

A closer integration of RE with the development process may enable the requirements to be better aligned and coordinated with other software development activities. This is the approach taken in agile software development to address the challenges of an increased rate of requirements change and the need for rapid software delivery [21]. In agile development, requirements are defined iteratively and in close cooperation within cross-functional teams [15] thereby supporting improved development efficiency and effectiveness, e.g. by avoiding the waste caused by developing and testing software based on unrealistic or unclear requirements. However, there are also challenges and risks with agile software development [2, 15] thus indicating that there are more factors at work. This research aims at increasing the insight into these factors and at enabling projects and organizations to select and configure a suitable level of RE integration for their development process that will enhance development efficiency and effectiveness.

Our main research question is if, and in which contexts, integrated RE can enable increased development efficiency and effectiveness by improving the coordination and alignment of requirements with software development. Based on an initial theoretical framework, we have identified two potential research tracks. Namely, (1) a gap finder that can assess the level of RE integration of an organization or project, and (2) a technique for integrating requirements documentation with code and test cases stored in the development environment. The purpose of the gap finder is to identify areas of weak RE integration and thereby enable improving the integration in those areas. Integrating the requirements documentation with the development environment will bring the requirements closer to both the automatic test cases and the source code, and the roles working with these artefacts. This technique may address several RE challenges concerning communication with development roles and with providing requirements specifications that correctly reflect the agreed requirements. In addition, the initial theoretical framework will be developed and validated against empirical data.

The rest of this paper is organized as follows. Related work is described in Section 2. Section 3 outlines the planned research approach, while Section 4 describes the research method and the current status. Finally, Section 5 summarizes this abstract.

2   Related work

Damian et al. found that RE can support increased effectiveness of software development and augment the efficiency and productivity of other processes and lead to improvements in, e.g. project planning, managing feature creep, testing, defects, rework, and product quality [5]. RE efficiency has been considered in research and some methods for improving the efficiency have been proposed for, e.g. requirements communications, negotiation and prioritization [7]. However, the bulk of the related research focuses mainly on efficiency of the actual RE activities, rather than on increased efficiency of the software development as a consequence of effective RE.

For example, only two of six papers at the 1st international workshop on RE efficiency [7] seem to focus on increasing the efficiency of the overall development process.

Concurrent engineering [12] is an approach to product development where engineering processes are carried out concurrently with extensive feedback and iteration [21] and where the developers are to consider all aspects of the development cycle from requirements to cost and quality. The gains reported for concurrent engineering include increased efficiency, productivity and quality, and reduced waste and shortened lead times [12]. Agile software development applies a concurrent approach by integrating the processes for requirements, design and implementation and the claimed gains are similar to those for concurrent engineering, including increased responsiveness to change [21]. Six industrial RE practices used in agile development and seven challenges connected to these have been identified by Ramesh et al. [15]. Although face-to-face communication over documentation is one of the principles of agile development, weak communication within agile projects, e.g. with the customer and at the project level, has been found to lead to challenges with cost and project-level schedule estimations and customer participation [15]. In addition, these gaps in communication, in combination with the minimal amount of documentation produced in agile development projects, have been reported to cause problems with scaling and evolving the software and with including new project members [15].

Alignment of RE with testing has been investigated by e.g. Sabaliauskaute et al. and Uusitalo et al.. Issues related to organization, process, people, tools, requirement changes, traceability and measurements have been reported to cause challenges in aligning requirements and testing for large-scale software development [17]. Furthermore, a number of industrial practices for supporting alignment of requirements and testing have been reported by Uusitalo et al.. These practices include traceability between requirements and test cases [23], as well as, increased communication between roles [23], e.g. by involving testers early in the project and in requirement reviews, and by establishing communication between testers and requirement owners. Similarly, Marczak et al. found that in requirements-driven collaboration there is often close communication between requirements and testing roles; key roles which when absent cause disruptions within the development team [13]. Furthermore, Stapel et al. found that most problems in global software development are related to communication, missing context, awareness and missing document information [22]. To address these issues, awareness of the communication paths was suggested, as well as, having 'ambassadors' physically present at the different sites to improve the information transfer [22].

Various techniques and methods for aligning and integrating the requirements specification with other development artefacts have been proposed. A lot of research has been done on supporting traceability [8] within requirements and between requirements and test cases. Another alternative approach is model-based development [9] where the requirements are described in a formal language from which code and/or test cases are then generated. Post et. al propose a more high-level approach where test cases are linked to formalized scenarios of the requirements [14].
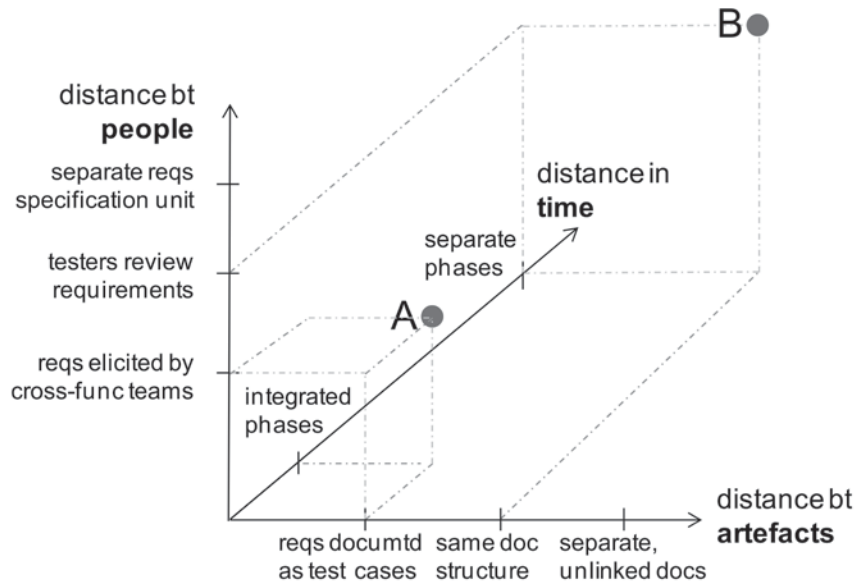
Fig. 1. A framework for integrated RE. Axes illustrated by requirements and testing alignment practices, placed in order of relative distance between roles, artefacts and in time. RE integration coordinates displayed for an agile project (A) and a waterfall project (B).

In agile development, integration of the requirements and testing artefacts is supported by behaviour-driven development (BDD). In this approach, the focus is on achieving an executable specification of the system by defining requirements as test cases. A domain-specific language (DSL) containing terms from the business domain is used to define the test cases. The DSL provides the customers and developers with a common language that reduces ambiguities and misunderstandings. Solis and Wang reviewed the available BDD literature and a number of BDD toolkits and found that the area is still under development [20]. In addition, the toolkits are limited to only supporting the development phase and do not provide the possibility to add domain-specific concepts to the DSL [20].

3 Research Approach

An initial literature study and an interview study into the alignment of RE and testing have been performed to gain a deeper insight into integrated RE. Based on these studies, a theoretical framework for integrated RE has been outlined. The framework consists of three dimensions, namely

- people: the distance between organizational units, roles, and individuals,
- artefacts: the distance in navigation and consistency between related documents,
- time: the temporal distance between related activities.

The initial version of the framework is depicted in Figure 1. On the axes, the relative distances caused by some practices are shown and the coordinates of an agile project (A) and a more traditional project (B) are shown. The agile project is positioned closer to origo, indicating that the RE integration level is higher for A than for the traditional project B.

Preliminary results from our case studies [1, 2, 3] and from related work [4, 5, 8, 11, 12, 13, 14, 17, 21, 22, 23] indicate that RE alignment and coordination with other development activities can be achieved by decreasing the distance to RE over one or more of the identified dimensions, thereby increasing the degree of integration. We suggest that the level of RE integration can be gauged by the distance between RE and other disciplines, where a shorter distance indicates a higher degree of integration. Large distances along one or more dimensions are assumed to cause weak alignment and coordination. These distances may be decreased by applying alignment practices. For example, for the dimension of people the distance between requirements engineers and testers may be decreased by including testers in requirements reviews; an identified requirements and testing alignment practice with the potential to improve test efficiency and effectiveness [23]. Furthermore, by assessing and adjusting the level of RE integration for a project or an organization, the development process may be optimized for the characteristics of a specific project or domain, e.g. project size, rate of requirements volatility etc.. For example, a large project most likely has many roles and different people involved in the requirements flow from customer to development, i.e. a large distance between people. While a small project might allow the customer to speak directly to the development team, i.e. have a short distance between people. For large projects, the integration may be improved by decreasing the distances over one or more dimensions. For example, by documenting requirements as test cases (decreasing the distance between artefacts) or by introducing cross-functional development teams including customer proxies (an agile RE practice which would decrease the distance between people). These practices would also decrease the distance in time between requirements and test definition, i.e. increase the integration also along the time axis.

4 Research Method

We intend to further detail and develop the theoretical framework (described in Section 3) by analyzing empirical data and comparing it to the framework, thereby extending or modifying the theory by constant comparison to the data [19]. This will then provide a theoretical basis for future research towards our two main research goals, i.e. (1) an assessment method for RE integration and (2) a technique for storing requirements in an integrated fashion with the source code and test cases.

Gap finder: a method for assessing RE integration. Industrial case studies [16] have provided us with rich insight into factors involved in the interaction between RE and software development for some RE challenges, primarily overscoping [1] and communication [3]. We intend to use these empirical results in designing a method for assessing the level of RE integration, by applying *causal-based modelling.* This

modelling technique can support reasoning about and assessing relationships in software engineering processes and thereby be used to identify software improvements [6]. We are currently complementing our previous case studies with an investigation into the situation and causal relationships for an agile development process. This case study also includes designing a set of integration metrics. The identification of causal relationships through qualitative methods will be combined with these quantitative metrics, and patterns between the two might be found. These patterns may then be used in the design of a method for identifying RE integration gaps, e.g. that the RE communication is weak between certain roles.

Integrating requirements in the development environment. As a first step towards a technique for integrating RE with source code, support for documenting and tracing requirements in an industrial development environment is planned to be prototyped. The technique will bring the requirements closer to the source code and the test cases (decreasing the artefact distance), and thereby also to the development engineers (decreasing the people distance). This proximity of the requirements documentation to development roles and to executable artefacts may close several communication gaps. In addition, the requirements documentation will be stored under the same CM control as the source code and automatic test cases. This CM control of requirements will support managing requirements, source code and test case versions in a uniform way. Furthermore, this technique can enable generation of requirements specifications that include information about the test execution results. For specific software builds, features and individual requirements that are not correctly supported can be identified through the specification, thus, providing increased visibility of software status from a requirements perspective. The use of domain-specific languages (DSL) to specify requirements including the relationships between them will be considered in a second step. Full-fledged usage of a DSL in the elicitation phase by supporting extending the DSL with domain concepts can be investigated either by extending an existing DSL or by designing a new one.

4.1 Evaluation of Research Results

This research is performed in collaboration with industrial partners using an empirical approach. The methods under development are, thus, evaluated already at the design stage with company representatives, i.e. through desktop evaluation [24]. In addition, collaboration with industry ensures that the problems addressed are real-life problems for which there are interested 'customers'. Furthermore, the designed methods will be tried out and empirically evaluated at one or more partner company.

4.2 Current status & plans

This research was initiated in Q1'10. The aim for 2012 is to design and empirical evaluate an RE integration gap finder and a prototype implementation of integrating requirements documentation with source code in a development environment. The design, implementation and main data gathering would then be performed during 2012, with the goal to analyze, report results and complete a Ph.D. thesis in 2013.

5   Summary

Integrating RE with other software development activities is an approach applied in agile software development to enable increased responsiveness to customer and market changes. Agile development is reported to mitigate some RE risks [15] and is often claimed to enable efficient development. However, agile RE practices have been found to pose new challenges and risks [15]. Several of these risks are connected to weak communication, despite face-to-face communication being an important principle of agile development. Further research is needed to support improved coordination and communication of RE, which is an important factor in enabling RE to support efficient development [1, 3, 5, 11, 22].

This research aims at increasing the efficiency and effectiveness of software development by providing support for customizing the level of RE integration with other software development disciplines for the specific project context, e.g. project size, frequency of requirements changes, domain etc.. A method for assessing the level of RE integration and identifying gaps and weak RE integration is planned to be developed. This method will be partly based on causal relationships identified through industrial case studies into RE challenges both in a phase-based process [1, 3] and in an agile development process. In addition, we plan to prototype and evaluate a technique for integrating requirements in a software development environment, thus documenting the requirements in the same system as the source code and the test cases. Our research is performed in collaboration with industrial partners and the methods that are developed will be empirically evaluated in an industrial setting.

References

1. Bjarnason, E., Wnuk, K., Regnell, B.: Overscoping: Reasons and Consequences – A Case Study in Decision Making in Software Product Management. Proc. of 4th Int. IEEE Workshop on Software Product Management, IWSPM'10, 30-39. (2010)
2. Bjarnason, E., Wnuk, K., Regnell, B.: A Case Study on Benefits and Side-Effects of Agile Practices in Large-Scale Requirements Engineering. Proc of 1st Int. Workshop on Agile RE, Lancaster, UK. (2011)
3. Bjarnason, E., Wnuk, K., Regnell, B.: Requirements are Slipping Through the Gaps – A Case Study on Cause & Effects of Communication Gaps in Large-Scale Software Development. Proc. of 19th IEEE Int Requirements Engineering Conf. (2011)
4. Curtis, B., Krasner, H., Iscoe, N.: A Field Study of the Software Design Process for Large Systems. Commun. ACM, vol. Nov. 1988, 1268-1287. (1988)
5. Damian, D., Chisan, J.: An Empirical Study of the Complex Relationships between Requirements Engineering Processes and Other Processes that Lead to Payoffs in Productivity, Quality, and Risk Management. IEEE Transactions on Software Engineering, vol 43, no 7, pp 433-453. (2006)

6. Dumke, R., Richter, K., Georgieva, K., Asfoura, E.: Process Improvement Based on Causal Networks, 8th ACIS Int. Conf Softw. Eng. Research, Manage.and Appl, pp.285-291. (2010)
7. Fricker, S., Seyff, N.: 1st Int. Requirements Engineering Efficiency Workshop – REEW 2011, ACM SIGSOFT Software Engineering Notes, Vol. 36, issue 3, pp. 26-28. (2011)
8. Gotel, O.C.Z., Finkelstein, C.W.: An Analysis of the Requirements Traceability Problem. 1st Int Conf on Requirements Engineering, pp 94-101. (1994)
9. Hasling, B., Goetz, H., Beetz, K.: Model Based Testing of System Requirements using UML Use Case Models. 1st Int Conf on Softw. Testing, Verif. and Valid., pp 367-376. (2008)
10. Karlsson, L., Dahlstedt, A. G., Regnell, B., Natt och Dag, J., Persson, A.: Requirements Engineering Challenges in Market-Driven Software Development-An Interview Study with Practitioners. Information and Software Technology, Vol. 49, issue 6, pp 588-604. (2007)
11. Kraut, R.E., Streeter, L.: Coordination in Software Development. Communications of the ACM, vol. 38, no. 3, 69--81. (1995)
12. Lawson, M., Karandikar, H. M.: A Survey of Concurrent Engineering. Concurrent Engineering 1994 2:1 DOI: 10.1177/1063293X9400200101 (1994)
13. Marczak, S., Damian, D.: How Interaction between Roles Shapes the Communication Structure in Requirements-Driven Collaboration. 19th IEEE Int Requirements Engineering Conf. (2011)
14. Post, H., Sinz, C., Merz, F., Gorges, T., Kropf, T.: Linking Functional Requirements and Software Verification. 17th Int. Conf on Requirements Engineering, 295-302. (2009)
15. Ramesh, B., Cao, L., Baskerville, R.: Agile requirements engineering practices and challenges: an empirical study. Inform. Systems Journal, vol 20, issue 5, 449-280. (2010)
16. Robson, C.: Real World Research. Blackwell Publishing. (2002)
17. Sabaliauskaite, G., Loconsole, A., Engstrom, E., Unterkalmsteiner, M., Regnell, B., Runeson, P., Gorschek, T., Feldt, R.: Challenges in Aligning Requirements Engineering and Verification in a Large-Scale Industrial Context. 16th Int Working Conf on Requirements Eng. Foundation for Software Quality (REFSQ), pp. 128-142. (2010)
18. Sawyer, P.: Packaged software: Challenges for RE. Proc. of the 6th Int. Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'2000). (2000)
19. Seaman, C.B.: Qualitative Methods in Empirical Studies of Software Engineering. IEEE Transactions on Software Engineering, vol. 25, issue 4, pp 557-572. (1999)
20. Solis, C., Wang, X.: A Study of the Characteristics of Behaviour Driven Development, 37th EUROMICRO Conf. on Softw.Eng. and Advanced Applications (SEAA), pp.383-387. (2011)
21. Sommerville, I.: Integrated Requirements Engineering: A Tutorial. IEEE Software, Vol. 22, issue 1, 16-23. (2005)
22. Stapel, K., Knauss, E., Schneider, K.: Using FLOW to Improve Communication of Requirements in Globally Distributed Software Projects. IEEE Proc. Int. Workshop on Collab. and Intercult. Issues on Req.: Comm. Understanding and Softskills. pp. 5-14. (2009)
23. Uusitalo, E.J., Komssi, M., Kauppinen, M. et al.: Linking Requirements and Testing in Practice. 16th IEEE Int Requirements Engineering Conf, NJ, USA, 265-270. (2008)
24. Wohlin, C., Gustavsson, A., Höst, M., Mattsson, C.: A Framework for Technology Introduction in Software Organizations. Proc. Softw. Process Improve. Conf., 167-176, Brighton, UK. (1996)

Traceability between System Model, Project Model and Source Code

Alexander Delater, Barbara Paech

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

Abstract. Traceability from source code to system model elements like requirements has been extensively researched. Even though existing approaches use various heuristics and methods to compute traceability links automatically, they do not return very satisfying and dependable results. In contrast to these approaches, we not only consider the system model, but also the project model, which is used for planning and organization in software development projects. In this thesis, we plan to create and utilize traceability links between elements from system model, project model and source code. We believe that by using elements from the project model as mediator connectors, links between elements from the system model and source code can be easily created. In this paper, we present the research problems that need to be solved as well as our principal solution ideas to tackle these problems.

Keywords: traceability, system model, project model, source code

1 Introduction

The software development process relies on traceability information captured throughout the evolution of a software product. Traceability supports, amongst others, program comprehension, change management, software maintenance, software reuse and prevention of misunderstandings [10]. Traceability between requirements and source code has been extensively researched in the past years and much progress has been made in this field. Because the manual creation of traceability links between requirements and source code is cumbersome, error-prone, time consuming and complex [22], a major focus in research is on (semi-) automatic approaches. Existing (semi-) automatic approaches use various techniques, e.g. information retrieval, execution traces, static/dynamic analysis, subscription-based or rule-based link maintenance or combinations of them. Even though these approaches use different heuristics and methods to compute traceability links between requirements and source code, they do not return very satisfying and dependable results [22].

In software development projects, two different types of models are used for abstraction: the *system model* and *project model* [16]. Model elements from the

system model describe the system under construction, such as requirements, use cases, components or design documents. Model elements from the project model describe the on-going project, such as work items, the organizational structure, iterations or meetings (we use the term work item instead of task to avoid misunderstandings with the term task used in requirements engineering). These two models have already been integrated within a model called MUSE: Management-based Unified Software Engineering [16]. The MUSE model is implemented in the model-based CASE tool UNICASE [4].

While the MUSE model describes the system to be developed and its project management, it does not provide traceability to the source code. Furthermore, the MUSE model supports the manual creation of traceability links, but it does not support the automatic creation of traceability links.

In this thesis, we want to extend the MUSE model by a new *code model* to support traceability to the source code. We want to study the usage of traceability links between these three models, namely system model, project model and code model. Moreover, we want to present a (semi-) automatic approach for creating traceability links between these three models. These traceability links are expected to support various development activities, such as program comprehension, change management and software maintenance. We want to implement the extended MUSE model and the proposed approach for (semi-) automatic traceability link creation in UNICASE and evaluate it in various case studies.

This paper is structured as follows: Section 2 describes the research problems concerning this thesis. Section 3 presents the proposed solutions and discusses their novelty. Section 4 gives an overview about related work. Section 5 discusses the applied research methods. Our progress concludes the paper in Section 6.

2   Problems

There are various problems that need to be solved in order to link source code with elements from system model and project model.

P1-Representations of Source Code: A problem is to define the representations of source code that make up the elements of the code model.

P2-Capturing & Inferring Traceability Links: The manual creation of traceability links is cumbersome, error-prone, time consuming and complex. Thus, a (semi-) automatic approach for capturing traceability links between the three models is necessary. Support for direct navigation between elements of all three models is also required.

P3-Identifying Relevant Traceability Links: The approach for solving P2 might create a lot of links. Support for the derivation of the most relevant links is necessary.

P4-Supporting Change Impact Analysis: With the different elements from the code model from P1 and approaches for capturing and identifying relevant traceability links from P2 and P3, several development activities can be supported. In this thesis, we want to focus on supporting change impact analysis and present an algorithm using the newly created traceability links.

3 Proposed Solutions

3.1 P1-Representations of Source Code

For the elements of the code model, we want to focus on file-based and change-based representations, because they are widely used in software development projects. For example, file-based representations are file resources containing source code or line(s) of code in these resources. Change-based representations are supported by a version control system (VCS), e.g. patch or revision/branch.

UNICASE is a plugin for the Eclipse integrated development environment (IDE). The Eclipse IDE supports various programming languages through additional plugins, e.g. Java, C++, Python etc. By integrating UNICASE and Eclipse with plugins for VCSs like Subversion [23] or Git [12], we can provide a comprehensive tool environment supporting the developers while they perform various development activities. By using these plugins, we can access file-based as well as change-based representations of source code.

3.2 P2-Capturing & Inferring Traceability Links

Work items represent a unit of work which describe changes to be performed to the code as well as new developments. They are the task descriptions used in many software development projects. As they are the basis of the daily work, they are regularly kept up-to-date [14]. Furthermore, as work items are used to describe pending work, they can also implicitly mention the relationships between system elements relevant to the current work item within its textual description, e.g. the requirement that needs to be implemented or a related design element.
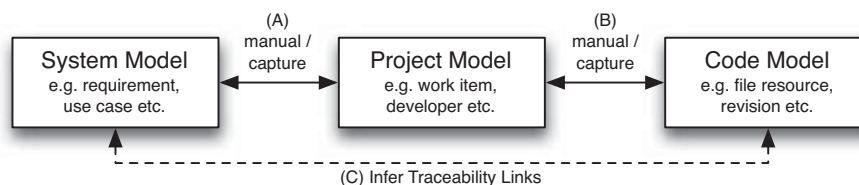
Fig. 1. Traceability between system model, project model and code model

We believe that by using project model elements as mediator connectors, traceability links between system model elements and code model elements can be easily created (see Fig. 1). The core idea of creating traceability links between elements of system-, project- and code model is letting the developers create these links themselves. First, the developer selects a work item and starts implementation. While working on the work item, all system elements (e.g. requirements, design documents) the developer looks at during implementation are automatically captured (see A in Fig. 1). After finishing the implementation of a

work item, the developer does not immediately commit the changes to the VCS. Instead, before the commit, s/he has to verify the list of captured traceability links. This means that the developer has to accept all or reject some traceability links that were captured. It is an open question whether traceability links could be suggested as likely to be relevant. This additional work results in very little overhead for the developers. After this verification, the newly created revision in the VCS is linked to the work item (see B in Fig. 1). It must be studied whether the set of links of one work item can be used efficiently to navigate between the elements linked to that work item, e.g. the requirements and the code related through that work item (see C in Fig. 1). Note that this approach also implicitly alleviates the problem of link maintenance, if it is assumed that any relevant change to a system element is performed only in context of a work item. The links of the most recent work items always provide the most up-to-date links between elements of the system model and code model.

3.3 P3-Identifying Relevant Traceability Links

The approach for capturing and inferring traceability links might create a lot of links. Support for the derivation of the most relevant links is necessary. We plan to implement an algorithm that provides a relevance ranking for each link based on the change history of the elements connected by the link. The change impact analysis can focus on the most relevant traceability links.

3.4 P4-Supporting Change Impact Analysis

We plan to implement an algorithm for change impact analysis using the most relevant captured and inferred traceability links. This algorithm bridges the gap between requirements and source code to answer questions as: What parts of the source code need to be changed based on a change in a requirement? We want to classify our new algorithm using the taxonomy presented by Lehnert [17] and compare it to existing algorithms. We expect that this algorithm is able to provide more detailed results during change management than existing algorithms.

4 Related Work

Maintaining traceability links between source code and other artifacts is a challenging task and therefore a field of intense research.

To the best of our knowledge, no approach uses work items to create traceability links between requirements and code. Either they create links between requirements and code using mostly (semi-) automatic approaches (e.g. information retrieval [1, 19, 20, 13, 6], execution-trace analysis [8, 11, 5], static/dynamic analysis [2], subscription-based or rule-based link maintenance [18] or combinations of them [7]) or only create links between work items and code [3].

Furthermore, other approaches only relate structures in the source code like classes, methods, lines of code or modules, files and resources to other artifacts like requirements [24]. This is also supported by our approach. However, our approach is also able to track exact changes in the source code.

An approach similar to ours for the automatic capturing of links was presented by Omoronyia et al. [21]. They have achieved traceability between use cases and source code. Their approach is based on tracing the operations carried out by a developer called navigation trails. However, this approach requires an elaborate model with rankings of navigation trails to derive the most relevant links. It is an open question whether the availability of work items can alleviate this ranking and how to define rankings for other elements, e.g. design documents touched while implementing a use case. Their approach is also able to identify which developer is involved in the realization of a specific use case. The contribution of Omoronyia et al. shows that tracking changes displays some advantages over the other approaches. For example, relating a developer to the source code and use cases is almost impossible with the other approaches, but very easy if changes/operations are tracked, like in our approach.

Except Omoronyia et al., all other approaches mentioned above try to create traceability links after the implementation of the source code. In comparison, our approach creates traceability links while the system is implemented. We track the changes made to the source code and link them to the work items they belong to. The work items themselves are linked to elements of the system model and new traceability links can be captured between them during development. Based on the intermediate work items, we expect to be able to infer reliable traceability links between system model elements and source code.

5   Research Methods

The overall goal is to validate our proposed solutions. To reach this goal, we apply a tool prototype driven approach where each conceptual research result is developed in parallel with a tool prototype based on the model-based CASE tool UNICASE. Thus, we are able to validate our results early by applying them in academic projects (e.g. bachelor/master theses), in practical courses as well as in the open source project UNICASE itself.

First case studies showed that links between system elements and project elements provide useful information for the work (by shortening the navigation paths of the developers) and that based on such links system elements are kept more up-to-date [15]. We want to conduct more case studies using our presented approach and developed tool support based on UNICASE.

For change impact analysis, traceability links can be evaluated by calculating two metrics: the percentage of actual matches that are found (recall) and the percentage of correct matches as a ratio to the total number of candidate links returned (precision). We want to apply these metrics to our algorithm for change impact analysis and compare the results to existing approaches, e.g. [1, 19, 20, 13, 6]. We want to compare the effort and quality of capturing traceability links

between requirements and source code of our presented approach to the results of other conducted exploratory experiments, e.g. by Egyed et al. [9].

6   Progress

In 2011, defined the representations of source code that we want to focus on (P1). Furthermore, we provided (semi-) automatic support for capturing traceability links between project model and code model. We used patches and revisions in a version control system as two possible types of representation of source code. Changes to the source code are tracked and when the developer commits some code changes, links between the code changes and the work item are captured (P2).

In 2012, we plan to provide (semi-) automatic support for capturing traceability links between system model and project model in UNICASE (P2). We will implement an algorithm that provides a relevance ranking for each link based on the change history of the elements connected by the link (P3). Furthermore, we plan to implement an algorithm for change impact analysis using the most relevant captured and inferred traceability links (P4). We will evaluate the algorithm using data from the open source project UNICASE. We expect to finish this thesis by mid 2013.

References

1. Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., Merlo, E.: Recovering traceability links between code and documentation, IEEE Transactions on Software Engineering, pp. 970-983 (2002)
2. Antoniol, G., Gueheneuc, Y.G.: Feature identification: A novel approach and a case study, In ICSM '05: Proceedings of the 21st IEEE International Conference on Software Maintenance, pp. 357-366 (2005)
3. Anvik, J., Storey, M.A.: Task articulation in software maintenance: Integrating source code annotations with an issue tracking system, In ICSM '08: IEEE International Conference on Software Maintenance, pp. 460-461 (2008)
4. Bruegge, B., Creighton, O., Helming, J., Koegel, M.: Unicase - an Ecosystem for Unified Software, In ICGSE '08: Distributed software development: methods and tools for risk management, ICGSE Workshop 2008 (Bangalore, India, 2008)
5. Burgstaller, B., Egyed, A.: Understanding where requirements are implemented, In 2010 IEEE International Conference on Software Maintenance, pp. 1-5 (2010)
6. De Lucia, A., Fasano, F., Oliveto, R., Tortora, G.: Recovering traceability links in software artifact management systems using information retrieval methods, Transactions on Software Engineering Methodology, vol. 16, no. 4, art. 13, ACM (2007)
7. Eaddy, M., Aho, A.V., Antoniol G., et al.: CERBERUS: Tracing requirements to source code using information retrieval, dynamic analysis, and program analysis, In the 16th IEEE International Conference on Program Comprehension (ICPC), pp. 53-62 (2008)
8. Egyed, A.: A Scenario-Driven Approach to Trace Dependency Analysis, Transactions on Software Engineering, vol. 29, no. 2, pp. 116-132, IEEE (2003)

9. Egyed, A., Graf, F., Grünbacher, P.: Effort and quality of recovering requirements-to-code traces: Two exploratory experiments, In RE '10: Proceedings of the 18th International IEEE Requirements Engineering Conference (RE) (2010)

10. Egyed, A., Grünbacher, P.: Supporting software understanding with automated requirements traceability, International Journal of Software Engineering and Knowledge Engineering, vol. 15, no. 5, pp. 783-810 (2005)

11. Eisenberg, A.D., De Volder, K.: Dynamic feature traces: Finding features in unfamiliar code (2005)

12. Git - Fast Version Control System. http://git-scm.com.

13. Hayes, J.H., Dekhtyar, A., Osborne, J.: Improving requirements tracing via information retrieval, International Conference on Requirements Engineering (2003)

14. Helming, J., Arndt, H., Hodaie, Z., Koegel, M., Narayan, N.: Semi-automatic assignment of work items, In ENASE '10, pp.149-158 (2010)

15. Helming, J., David, J., Koegel, M., Naughton, H.: Integrating system modeling with project management - a case study, In COMPSAC '09: Proceedings of the 2009 33rd Annual IEEE International Computer Software and Applications Conference (Washington, DC, USA, 2009), IEEE Computer Society, pp. 571-578 (2009)

16. Helming, J., Koegel, M., Naughton, H.: Towards traceability from project management to system models, In TEFSE '09: Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering, pp.11-15. IEEE Computer Society (2009)

17. Lehnert, S.: A taxonomy for software change impact analysis, In Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution (New York, NY, USA, 2011), IWPSE-EVOL '11, ACM, pp. 41-50 (2011)

18. Maeder, P., Gotel, O.: Towards Automated Traceability Maintenance, Journal of Systems and Software (2011)

19. Marcus, A., Maletic, J.I.: Recovering documentation-to-source-code traceability links using latent semantic indexing, In Proceedings of the 25th International Conference on Software Engineering, pp. 125-135. IEEE Computer Society (2003)

20. Marcus, A., Maletic, J.I., Sergeyev, A.: Recovery of traceability links between software documentation and source code, International Journal of Software Engineering and Knowledge Engineering, vol. 15, no. 5, pp. 811-836 (2005)

21. Omoronyia, I., Sindre, G., Roper M., Ferguson J., Wood, M.: Use case to source code traceability: The developer navigation viewpoint, In 2009 17th IEEE International Requirements Engineering Conference, pp. 237-242 (2009)

22. Spanoudakis, G., Zisman, A.: Software traceability: A roadmap, In Handbook of Software Engineering and Knowledge Engineering, World Scientific Publishing, pp. 395-428 (2004)

23. Apache Subversion. http://subversion.apache.org.

24. Treude, C., Storey, M.A.: How tagging helps bridge the gap between social and technical aspects in software development, In Proceedings of the 31st International Conference on Software Engineering (ICSE), pp. 12-22 (2009)

Engineering User Experience Requirements
An Incremental Approach

Pariya Kashfi

PhD Candidate
Software Engineering Division
Department of Computer Science and Engineering
Chalmers University of Technology
`pariya.kashfi@chalmers.se`

Abstract. Both functional and quality requirements should be considered in software development in order to result in a positive user experience. While requirement engineering has evolved and handles non-functional requirements, there is still a lack of methods and guidelines for practitioners to address quality requirements, particularly those requirement that are not related to performing a task or accomplishing a goal. One of the gaps in dealing with this type of requirements is that the suggested methods do not consider current software engineering practices, and are difficult to put into practice by practitioners. In this project, we propose an "incremental" approach to engineer user experience requirements. The key concept in an incremental approach is to discover barriers in current practices, and suggest efficient and cost-effective improvements.

1 Introduction

Many studies have pointed out the importance of taking both functional and non-functional, i.e. quality, requirements into account in software (SW) development, particularly in Requirements Engineering (RE). Nevertheless, there is still a lack of practical guidelines and methods for Software Engineering (SE) practitioners in dealing with quality requirements especially those quality requirements that are not related to performing a task or accomplishing a goal, such as emotional connection, joy, and excitement. We refer to this type of requirements as non-task-related requirements or user needs. The history of studying non-task-related user needs goes back to the 90s [1] when researchers initiated studies mostly under the name of *User eXperience* (UX) to deal with different types of user needs, particularly in the field of Interaction Design (ID) [2–4, 1]. Even so, within SE, there has been just a few studies that have taken UX into account.

To give proper support to SE practitioners regarding UX, i.e. taking various types of user needs into account in SW development, we consider the following steps necessary: (i) understanding the concept of UX and its composing elements (ii) having guidelines on how to improve current SE practices with minimal effort and cost in order to reach a positive UX in developed SW. So far, UX related research in neither ID nor SE have covered these steps. Studies in ID are mostly just theoretical contributions without any actual use in industry. On the other hand, in SE, studies either are merely theoretical

contributions or provide too narrow views on UX and do not cover all of its aspects as it is discussed in ID. This indicates a gap in UX theory as well as practice for SE.

To bridge this gap, in this project, we aim to investigate the current UX advancements in SE and ID from SE practitioners' perspective and based on that suggest an approach to improve current practices in order to develop SW with positive UX. We believe that a realistic approach toward UX in SE, particularly in the RE phase, is an *incremental* approach that does not require radical changes to the current practice, as opposed to a revolutionary approach. The key concept in such an approach is to discover barriers in the current SE practices, and suggest efficient and cost-effective improvements. Even though the approach should deal with all phases of SW development, because of time constraints, we initially focus on RE since engineering UX requirements is a key to develop SW with positive UX in later phases. The research question we aim to answer is: *How can SE practitioners be supported to develop SW with right and required UX?* Our research consists of three main steps (i) defining UX for practitioners (ii) discovering barriers to, and proposing improvements for developing SW with positive UX. This step also includes recommendation of methods for identifying the barriers (iii) proposing methods for evaluating the effects of applying the suggestions in the previous step on UX. We aim to reuse the existing theory rather than to develop it hence the contribution of this project will be mainly methodological rather than theoretical.

2 Related Work

People choose products or services over one another because they provide "what" they want to do, i.e. function, and also because those products or services are more preferable in terms of "how" they provide that function, what "messages" they communicate to the society, what "feelings" they trigger, etc. The needs, i.e. motivations to choose a product or service, include both functional and quality needs. Moreover, these needs are both task-related (e.g providing an email service, being easy to use) and non-task-related (e.g. emotional connection) [2]. It is important to consider various user needs in SW development since users' decisions to buy and use a SW are influenced by not only task-related, but also non-task-related needs [1, 2]. Hence, ultimate success of the developed SW as well as business goals such as the market share, profit and company image will be reached by satisfying various needs of users [5]. Task-related and non-task-related user needs have been in focus for more than two decades in ID [1]. Nevertheless, in SE, while RE has evolved enough to address part of task-related user needs via activities to elicit, identify and evaluate functional requirements, dealing with non-task-related quality requirements still is an open problem. In the following, a summary of the UX related advancements in ID as well as some of the few related studies in SE is presented.

2.1 User Experience in Interaction Design

The term *user experience* has been around for more than two decades. Donald Norman brought UX to wider knowledge in mid 1990s [1]. Hassenzahl et al. [6] define UX as "a consequence of a user's internal state (predispositions, expectations, needs, motivation,

mood, etc.), the characteristics of the designed system (e.g. complexity, purpose, usability, functionality, etc.) and the context (or the environment) within which the interaction occurs (e.g. organizational/social setting, meaningfulness of the activity, voluntariness of use, etc.)." The definition shows that UX deals with both task-related and non-task-related needs.

A UX framework, or model, consists of definitions of key UX elements, and their functional relations [2]. Some of these frameworks such as [2] are merely theoretical, and lack methodological contributions, empirical studies to support their results, or recommendations for development of SW with positive UX. On the other hand, in recent years, frameworks have been developed that complement their theoretical contributions with methodological and empirical results. For instance, in Mahlke's framework [4], along with defining UX, considering its various aspects, its composing elements and their relation, there is a collection of suggested methods to measure various aspects of UX. Moreover, Mahlke reports empirical data to support his contributions. Similarly, Zimmermann's framework [1] includes both theoretical and empirical results. However, Zimmermann focuses on developing two new methods for measuring only two aspects of UX and has a narrower view compared to Mahlke.

From SE practitioners' perspective there are shortcomings associated with these frameworks: (i) these frameworks are theoretical rather than practical (ii) they are developed based on a design and creation, as in ID, rather than an engineering, as in SE, perspective (iii) they are high level, complex and abstract (iv) they are not easy to understand for someone with a SE background (even in cases with methodological contributions) (v) they lack guidance on how their construct of UX can be applied in current SW development practices.

2.2   User Experience in SE

A related concept to UX in SE is the Quality in Use (QiU) model defined in ISO25010 [7] (an extension to ISO9126 [8]). According to ISO25010, QiU is "the degree to which a product or system can be used by specific users to meet their needs to achieve specific goals with effectiveness, efficiency, freedom from risk and satisfaction in specific contexts of use." Even though the QiU model does not include any definition of UX or any direct reference to that concept, it has been applied in the context of UX by SE researchers such as Doerr et al. [9].

Doerr et al. tried to discover the relationship between the Internal and External Qualities (I&EQ) (the quality attributes introduced in ISO9126 [8]) – i.e. functionality, reliability, usability, efficiency, maintainability, and portability– and UX that in their view was equivalent to QiU. Beside the QiU model, the Doerr's UX model was influenced by the *Technology Acceptance Model* (TAM) [10]. The TAM model is considered to be the most widely used theoretical model in the Information System (IS) discipline [11]. In this model, *perceived usefulness* and *perceived ease of use* are considered as the fundamental determinants of system use. Perceived usefulness concerns the task and goal related aspect of using a system, while perceived ease of use concerns the effort of using the application. While we appreciate Doerr et al.'s effort in opening a window to UX in SE, and mapping the existing SE terminologies to the less explored concept of UX, we find this study suffering from a weak theoretical basis, in particular, we do not agree

to their definition of UX. Doerr et al.'s findings on correlation between I&EQ and their construct of UX cover only the task-oriented aspects of UX. Moreover, they have only relied on those I&EQ that can be measured objectively and via questionnaires. While this is an interesting step toward influencing some aspects of UX, we believe that the model could be improved by applying the existing UX frameworks, and extending the study to other aspects of UX, such as emotions, to support SE practitioners in those aspects as well.

In a later study [12], Doerr et al. focused on measuring the future user satisfaction in early stages of the SW development life cycle, i.e. the RE phase. They suggested using a standard user satisfaction measurement tool, i.e. questionnaire, to prioritize the product features and improve the product's UX . In addition, they proposed a SW quality model named AMUSE (Appraisal and Measurement of User Satisfaction). The model was influenced by the QiU model in ISO9126 [8], and TAM. This study includes recommendations for tools to be used for measurement of user satisfaction. Nevertheless, the AMUSE model is limited to effectiveness and productivity, i.e. the task-oriented aspects of user satisfaction.

Before Doerr et al. [12], the idea of considering UX in RE was studied by Bentley et al. [13]. Where they suggested formally addressing *emotional requirements* that often have been neglected in RE. Bentley et al. suggested improving the requirements elicitation methods to support emotion-related requirements. They provided no insight into elicitation and documentation of these requirements, and the question of how these requirements should be incorporated and included in established RE techniques remained an open problem. Moreover, the study was limited to investigating three theories known to contribute to computer game enjoyment, and no empirical data was provided to support whether these theories actually enhance UX in SW in general.

The concept of emotional requirements has been discussed by other researchers such as [14–16] as well. Moreover, Callele et el. in [15] introduced the concept of *experience requirements* with an aim to use traditional RE techniques to improve the game development practices. Beside emotions, other related concepts to UX in SE are values, motivations, belief, and hedonic aspects of SW. Thew and Sutcliffe [17] presented a method called VBRE (Value Based Requirements Engineering) in order to support understanding and dealing with soft issues in RE such as stakeholders' emotions, values, and motivations. One advantage of the VBRE method is that it can be integrated to the current elicitation activities in RE. Thew et al. have also suggested the VBRE taxonomy, a taxonomy of values and their consequences on the SW development process and design. A section of this taxonomy includes emotions. We find such a taxonomy useful in guiding the elicitation process. The emotion taxonomy can be extended to include more issues related to emotional requirements and UX in general. The concepts of emotion, value, and belief and their importance in SW development is discussed by Ramos et al. [18] as well. Finally, Nass et al. in [19] emphasized the importance of finding the right balance between functional and hedonic aspects of SW. They presented a SW development approach to bridge between business and user goals. Their approach relies on task-oriented RE [20].

Another related study in the area of UX in SE is the FUN project [21]. The project is based on a quality model called e4FUN [22]. This model emphasizes on joy-of-use

from a cognitive behavioral perspective, avoiding the subjectivity of experience. The main focus of FUN is on pattern-based approaches in developing SW with positive UX. The results of the project include interaction patterns called "Fun Patterns", that can be integrated into the SW development process. One of the results of FUN is KREA-FUN [22], with a more general contribution to SE, discussed below.

In 2007, Kerkow et. al introduced a systematic approach to improve the joy-of-use in SW products [22]. This approach is realized in form of a workshop named KREA-FUN that aims to elicit and identify creative ideas of how to design for joy. In this workshop, domain experts, users, SW engineers, developers, managers, support personnel, and training personnel sit together to investigate and improve the joy-of-use in a SW. The authors refer to the process of eliciting and identifying ideas for designing for joy-of use as *engineering-joy-of-use*. The study includes specific methods and guidelines to add to the current practice in order to reach a better UX. In our view, SE practitioners will benefit from methodological contributions such as KREA-FUN. Still, such contributions can be improved by providing guidance on evaluating the effect of applying the proposed methods, and approaches to motivate application of those methods in the SW organization. Moreover, we view this approach as an additive rather than an incremental approach.

From other related studies in SE, we can refer to those discussing various approaches such as goal-oriented RE [23], scenario-based RE [24], and application of ID methods in SW development [25, 26]. These studies usually aim to improve usability or user acceptance, and do not go beyond task-related aspects of SW quality. Hence, they do not directly deal with UX.

In summary, SE researchers approach UX from an engineering perspective [5]. They try to find some physical and objective elements that influence UX and therefore make it possible to consider UX in SE activities [5]. In SE, UX is treated the same as other quality characteristics. While providing an opportunity to benefit from the existing advice on how to correctly measure various quality characteristics in SE, and to some extent influencing UX during SW development, this approach leads to a narrow view on UX since not all aspect of UX has so far been covered in the existing SW quality models. Additionally, similar to UX frameworks, one shortcoming of the studies in SE is that these studies not always provide empirical data to support their findings, and also there is a lack of guidance on how to integrate these methods in current SE practices.

2.3 Summary

In conclusion, there is limited theory and practice concerning how SE, particularly RE, practitioners should address UX. The UX frameworks lack practical guidelines and methods for practitioners. On the other hand, the engineering approaches toward UX suffer from not taking the whole aspects of UX into account and covering merely the usability, i.e. task-related, aspect of UX. Additionally, these approaches lack empirical data to support their results and taking current SE practice into account in providing suggestions and guidelines for practitioners. What a SE practitioner needs is not yet another theoretical UX contribution but empirical data to support the current theoretical and methodological contributions, and suggestions on how to improve the current practice, as well as how to remove the current barriers in SW development activities.

3 Research Strategy

In this project, we aim to familiarize RE practitioners with what they should take away, do differently, or add to their current practice in order to reach a better UX in developed SW. We focus on developing an incremental rather than a revolutionary approach toward UX. Our research will focus on the intersection of two fields – SE and ID. The goal is not to create a new UX framework, but to build on the existing advancements in both SE and ID. Therefore, our contributions will be mainly methodological rather than theoretical. Our research project includes three main steps followed by evaluation of the research findings. Literature review, case study, and post-hoc analysis are the methods we will apply in this study. In the following, we present our research strategy in more details.

3.1 Step One: Defining UX for SE Practitioners

The first step is to provide an engineering definition for UX. This does not mean that we necessarily should present a new definition of UX. We try to refine one of the current definitions. For this purpose, we will review the related literature to find the existing definitions.

3.2 Step Two: Discovering Barriers and Proposing Improvements

This step consists of three sub-steps: (i) developing and describing methods to discover barriers to developing SW with positive UX in current SE practices (ii) discovering barriers using the proposed methods. In particular, we are interested in high-level barriers, for instance how the SW development organization communicates with the end users (iii) proposing improvements to current practices based on the results from the previous sub-step. One main result of step two is guidance on generating UX-related requirements specification, i.e. describing the UX requirements in a simple and usable form. For the purpose of this step, an industrial practice will be chosen as a case study to investigate the current practice, and as a basis to discover barriers and suggest improvements.

3.3 Step Three: Proposing Methods for Evaluating UX-effects

It is important to be able to evaluate the effects of the previous steps on UX, so we should propose methods to evaluate the effects of changes in the practice on UX. In this step, we perform a post-hoc analysis of an already developed SW, preferably in the same industrial practice chosen as the case study for step two.

3.4 Evaluating The Results

To evaluate and refine the findings of the project, a workshop will be held for a group of SE practitioners. The workshop will be followed by a questionnaire to gather comments and ideas on the results.

References

1. Zimmermann, P.G.: Beyond Usability–Measuring Aspects of User Experience. PhD thesis (2008)
2. Hassenzahl, M.: The thing and I: understanding the relationship between user and product. Funology: from usability to enjoyment (2003) 31–42
3. Jordan, P.W.: Designing Pleasurable Products: An Introduction to New Human Factors. Taylor & Francis (2000)
4. Mahlke, S.: User experience of interaction with technical systems. PhD thesis, Berlin, Technical university (2008)
5. Kerkow, D.: Don 't have to know what it is like to be a bat to build a radar reflector-Functionalism in UX. In Law, E., Vermeeren, A.P., Hassenzahl, M., Blythe, M., eds.: Towards a UX manifesto, COST294–MAUSE affiliated workshop. (2007) 19–25
6. Hassenzahl, M., Tractinsky, N.: User experience – a research agenda. Behaviour & Information Technology **25**(2) (March 2006) 91–97
7. ISO25010: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. Volume 2011. International Organization for Standardization, Geneva, Swiss (2011)
8. ISO9126: Software engineering - Product quality. Technical report, International Organization for Standardization, Geneva, Swiss (2001)
9. Doerr, J., Kerkow, D.: Total control of User Experience in Software Development – a Software Engineering dream? In Law, E., Hvannberg, E., Hassenzahl, M., eds.: Proceedings of the The Second COST294MAUSE International Open Workshop User ExperienceTowards a Unified View. (2006) 94–99
10. Davis, F.D.: Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Quarterly **13**(3) (September 1989) 319–340
11. Lee, Y., Kozar, K.A., Larsen, K.R.T.: THE TECHNOLOGY ACCEPTANCE MODEL : PAST , PRESENT , AND FUTURE. Communications of the Association for Information Systems **12**(1) (2003) 752–780
12. Doerr, J., Hartkopf, S., Kerkow, D., Landmann, D., Amthor, P.: Built–in User Satisfaction – Feature Appraisal and Prioritization with AMUSE. 15th IEEE International Requirements Engineering Conference (RE 2007) (October 2007) 101–110
13. Bentley, T., Johnston, L., von Baggo, K.: Putting some emotion into requirements engineering. In: Proceedings of the 7th Australian Workshop on Requirements Engineering, Citeseer (2002) 227–241
14. Callele, D., Neufeld, E., Schneider, K.: Emotional Requirements. IEEE Software **25**(1) (January 2008) 43–45
15. Callele, D., Neufeld, E., Schneider, K.: An Introduction to Experience Requirements. 2010 18th IEEE International Requirements Engineering Conference (September 2010) 395–396
16. Maiden, N.: Requirements and Aesthetics. IEEE Software **28**(3) (May 2011) 20–21
17. Thew, S., Sutcliffe, A.: Elicitation of Values, Motivations and Emotions: The VBRE Method. In: 1 st International Workshop on Values in Design - Building Bridges between RE , HCI and Ethics Table of Contents, Lisbon, Portugal (2011)
18. Ramos, I., Berry, D.M., Carvalho, J.a.A.: The Role of Emotion, Values, and Beliefs in the Construction of Innovative Work Realities. In: Proceedings of the First International Conference on Computing in an Imperfect World. Soft-Ware 2002, London, UK, UK, Springer-Verlag (2002) 300–314
19. Nass, C., Adam, S., Doerr, J., Trapp, M.: Balancing User and Business Goals in Software Development to Generate Positive User Experience. Volume 396 of Studies in Computational Intelligence. Springer Berlin / Heidelberg (2012) 29–53

20. Adam, S., Doerr, J., Eisenbarth, M., Gross, A.: Using Task-oriented Requirements Engineering in Different Domains Experience with Application in Reseach and Industry. In: 17th IEEE International Requirements Engineering Conference, IEEE (August 2009) 267–272

21. Fun project. http://fun-of-use.org (2006)

22. Kerkow, D., Graf, C.: KREA-FUN : Systematic Creativity for Enjoyable Software Applications. In: FUN 2007 Proceedings: Workshop for Design Principles for Software That Engages Its Users and Facing Emotions: Responsible Experimential Design. (2007)

23. Bolchini, D., Mylopoulos, J.: From Task-Oriented to Goal-Oriented Web Requirements Analysis. In: Proceedings of the Fourth International Conference on Web Information Systems Engineering. WISE '03, Washington, DC, USA, IEEE Computer Society (2003) 166—-

24. Sutcliffe, A., Maiden, N., Minocha, S., Manuel, D.: Supporting scenario-based requirements engineering. IEEE Transactions on Software Engineering **24**(12) (1998) 1072–1088

25. Sutcliffe, A., Thew, S., Jarvis, P.: Experience with user–centred requirements engineering. Requirements Engineering **16**(4) (May 2011) 267–280

26. Seffah, A., Jan, G., Desmarais, M.: Why HCI and Software Engineering Integration? (2010) https://securedoc.gi.polymtl.ca/mdesmarais/CHISE/reference.shtml.

Method for the Conceptual Phase of an Integrated Product and Service Design Applied to Construction Project

Cyril MAUGER[1, 2]

[1]Department of Service Science & Innovation, Public Research Centre Henri Tudor, Luxembourg, Luxembourg
[2]Laboratoire de Conception, Fabrication Commande, Arts et Métiers ParisTech, Metz, France
cyril.mauger@tudor.lu

Abstract. Requirements Engineering has largely contributed to the improvement of the Software domain concerning client's satisfaction. This proposal aims to show how it could also contribute to improve the Construction domain. The first thing concerns the identification and definition of a taxonomy based on the Requirements Engineering. The second one aims to contribute to the emergence of the Requirements Engineering discipline in the Construction domain. The solution presented here is based on the identification of similarities at several levels between these two domains, including issues about client's satisfaction. This leads toward a knowledge transfer from Requirements Engineering to the Construction domain.

Keywords: Construction Sector, Architectural Programming, Conceptual Phase, Requirements Engineering, Architecture/Engineering/Construction

1   Introduction

The **architectural programming** (aka **briefing**) phase consists in defining the framework and expectations of a construction project through a statement of customers' requirements [1]. It precedes the design phase handled by the architect who draws possible plans from this statement. The requirements are gathered by the **contracting owner** who consults stakeholders and future users of the building. For the elaboration of the requirements document (i.e. the **brief**), the **programmer** who assists the contracting owner specifies the characteristics of the building to design. The specifications are then used by the architect to design a building that will meet the clients' requirements. The content of the requirements document are design parameters including space planning (e.g. surfaces, volumes, and spatial organisation) but also quality aspects (e.g. architectural expression, feeling, or beauty).

Current briefing practice is highly criticized by the **Architectural/Engineering/Construction** (AEC) research community who considers it as inadequate, not sufficiently explicit, and limited [2]. They tends to be solution-focused [3], provides description or rationale [4] of existing solutions rather than pure requirements [5, 6] and this before a thorough understanding of the client's requirements [7].

This reasoning on solutions leads to multiple problems identified in the AEC sector. Solution-oriented specifications over-constrain the architect leading to less innovation or creativity in his design when he does not completely ignore the brief (i.e. the demand). It also gives birth to a copy/paste phenomenon where existing solutions are reused to provide the final solution. Unfortunately, very often, such an outcome does not meet the implicit requirements hidden behind the solutions described by the client.

There is a need to change the way of thinking about a building from solutions to requirements, in a more abstract but measurable way. Existing guides or methods [8, 9] propose a succinct definition of what define a building but their underlying conceptual structure lacks from rigor and analysis. Concepts are fuzzy, incomplete, and quite independent whereas in reality, there are a lot of relationships between them. As a result, the traceability of the requirements is totally lost when design issues required modifications on the building's plan. Those concepts are directly related to the client's ill-defined and vague vocabulary (e.g. needs, wishes, constraints, objectives) compared to the architect's one which is based on precise definitions associated to physical solutions (e.g. net area, floor loads). There is a lack of definition for the concepts which lead to a confusion in the expression of the clients' demand. As a result, clients do not master their demand and are not able to correctly evaluate the offer, i.e. the proposition of the architects according to their requirements. Consequently, changes occur all the time during the production of architectural plans, increasing costs and delaying the ending. When the project is finally completed and built, and started to be used, the retained proposal does not meet the client's quality requirements and little can he do to control or even objectively contest the final result.

These observations led to the identification of a major bottleneck, presented for this Doctoral Symposium: How to change the architectural programming activity, from an ill-defined analysis activity producing over-specified briefing based on experience, to a requirement oriented activity based on engineering? AEC desperately lacked of theoretical foundations to clearly and completely describe a building system with high level abstract concepts rather than concrete solution objects. Other engineering domains (i.e. Software Engineering, Mechanical Engineering, and Industrial Engineering) are well supported on this point by well-tried methods (e.g. Requirements Engineering), approaches (e.g. KAOS, i*), and techniques (e.g. use case, class diagram). This paper is about how Requirements Engineering (RE) could help AEC building this foundation from its knowledge and experience?

2 Relevance

Quality of the offer is mainly driven by quality of the demand. This statement is true for software products concerning requirements [10]. In fact, the quality of the demand, i.e. quality of the requirements [11], could be identified as one of the causes of software engineering project failure regarding clients' satisfaction [12]. Its impact on the product development process in terms of costs and delays increases at each step [13]. This well-known issue revealed by Boehm and Papaccio for software engineering [13] is widely used in AEC project management to explain changes management issue. Bad definition of the architectural problem leads to changes costing each time more money and more time to correct. It shows that the interest of better defin-

ing the product or system to design could have major impact on quality, costs and delays results.

Providing a view of the concepts required for structuring the client requirements at the architectural programming step would guarantee a better integration of them in the design by the architect. Moreover, this would empower the client during the evaluation of the architects' proposal because knowing well what was demanded in first place allows him to better appreciate the final result. The result of my research would provide enough elements to the client for making a clever decision about the architects' proposal regarding his demand (i.e. compliance [14]).

The last benefit concerns the release of the architect creativity and innovation in the design while keeping focus on the clients' requirements. Architects are often afraid of the content of a brief and perceive it as an amount of insoluble constraints limiting their capacities. By providing a conceptual definition of a building, structured around concepts (e.g. requirements, needs, objectives), it would be easier to express true requirements rather than description of solutions. Moreover, such a structure could be implemented in a database or software that would help the architects dealing with this huge amount of information in a quicker and easier way during the design process [15].

3 Contribution

Two main issues are addressed in my research proposal. The first one concerns the identification and definition of a taxonomy for AEC practitioners i.e. a systematic classification of the concepts that are of interest for clients during the briefing phase. The second one consists in moving the current architectural programming practice from a solution-oriented perspective to a problem driven perspective. In short, in comparison with what happened in the software domain in the 80s, it aimed to contribute to the emergence of a RE discipline in the construction domain.
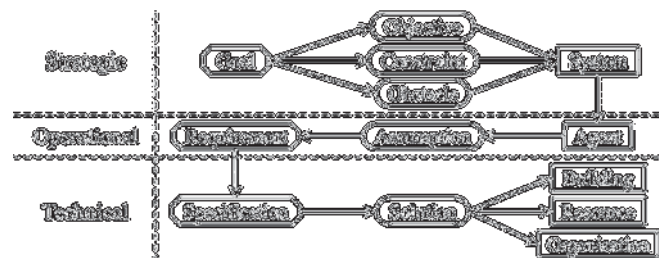
Fig. 1. First proposition of AEC taxonomy based on RE concepts

The preliminary taxonomy (**Fig. 1**) was structured around 3 levels: **strategic**, **operational**, and **technical**. The strategic level reflected the view and vocabulary of the client (cf. **Table 1**) about his business and the integration of the construction project in it. Starting from general **goals** (e.g. provide setting for education of 600 children), **concrete objectives** (e.g. keep the 600 children dry when it's raining) and **constraints** (e.g. parents cannot park inside the school) were defined. **Obstacles** (e.g. the 600 children need space to play when it is raining during breaks) could prevent goals'

achievement in a domain [16]. Based on this first set of concepts, a global **system** could be designed with a high level of description (e.g. a school, an education system).

At the operational level, the global system started to be defined from an external point of view (i.e. the system is a black box) beginning with elicitation of the **agents** (e.g. parents, children) that would interact or be involved with it. These agents were part of the system's environment. **Assumptions** (e.g. a child will not hurt another one if an adult can see him) were formulated about them to anticipate potential behaviour of the agents with or within the system. According to an objective or constraint (e.g. children must always feel safe at school) and assumptions about agents, **requirements** (e.g. a child should potentially always be seen by an adult inside and around the school) about the system were specified.

The system was considered as a white box at the technical level. Previous levels described what the system must do or be; this one developed how it would do it using architect's vocabulary. The defined requirements were then refined into **specifications** using domain knowledge [17]. A set of specifications could be achieved by a **solution**. Each solution was composed of internal components of the system and relation between them. Those components were classified into **building** (e.g. director's office), **organisation** (e.g. teaching or surveillance) and **resource** (e.g. teacher, white board).

All of these concepts are inter-related and allow going back and forth between each concept. Based on this RE classification, information about the building, the service and the business of the "building system" could be structured and prioritized in time and importance. Thus, this first taxonomy proposed a theoretical way to establish a traceability of the requirements in AEC and the specifications from the goals to the solutions of the architects. It would help programmers dealing with the amount of information and provide rationales to the clients for modifying architects' proposal to better meet his requirements. Nevertheless, in practice, this traceability is not that simple to put in place [14].

4 Analysis

Berry and Wieringa already identified that there are similarities between AEC and RE [5, 18, 19]. They concluded that RE could learn from AEC, but it is reciprocal: AEC could learn a lot from RE and that is the point of this PhD work. Berry in [18] proposed a set of similarities around the activities of build or remodel a house and develop or enhance a software. This document goes further in the identification of similarities that can complete his analysis. The roles in AEC and RE are quite similar, **Table 1** gives a more complete description than Berry [18] of the different engaged roles. The relationships between each role are the same in AEC and RE as shown in [18]. Issues concerning the definition of the artefact to design [19] and importance of this conceptual phase [6, 10] (i.e. architectural programming and requirements management) are also the same.

Another similarity concerned the type of deliverables. Three deliverables can be counted in AEC: the Statement of Needs (SoN, "préprogramme" in French), the Strategic Brief (SB) and the Design Brief (DB) [3]. In RE, this correspond to some RE

classification of deliverable into: the Customer Requirements Specification (CRS), the System Requirements Specification (SysRS), and the Software Requirements Specification (SRS) [20]. SoN and CRS contain a coarse description of the system expressed with original statements transferred verbally from the client to the brief writer (aka programmer), or directly provided to the requirements engineer (aka system analyst). SB and SysRS are more accurate in their content. They gather information, requirements required by the project team to establish a "real" project or system. And finally, DB and SRS are both developed from the previous deliverable level and specify the building or software. Building and software are only parts of the global project or system to design. Each deliverable gives more details about the artefact to design, from high level requirements to low level specifications.

Table 1. Similarities of roles between AEC and RE

Software Engineering	AEC	Description
Client	Contracting Owner	A person or organisation who pay for a system development
User/Agent	Operator/Customer	A person who uses the functionality provided by a system to deliver a product or service
Customer	User	A person or organisation who receives a product or service [20]
Requirements Engineer	Programmer	A person who deals with stakeholders' requirements and provides deliverables to designers
Developer	Architect	A person who designs a system to build from deliverables
Programmer	Builder/Contractor	A person who makes a system "real"

The first elements of analysis provide a vision of the global building system as composed of three parts (**Fig. 1**): a product part (i.e. the building), a service part (i.e. the organisation installed inside it) and a human part (i.e. people who work inside the building for the organisation). A state of the art on the different conceptual design approaches reveals that each domain is mainly focused on its own item/artefact and did not deal with the global system to develop.

In Software Engineering, the system to design is composed of three parts: a software part, a hardware part and a human part. It is true that RE is mainly focused on the software part. By analogy with a building system, this software part corresponds to its service part, and the hardware part to the building itself. Based on this quick analogy, the definition of a building or rather its service or organisation part would be done using concepts that defined software. These concepts were one of the missing elements in AEC to define properly a building system.

From the identification of these similarities between AEC and Software Engineering, an analysis of the RE literature and practices was led so as to identify potential contribution of each. RE and Goal-Oriented Requirements Engineering (GORE) were retained and studied to that extent. Even if their contribution was useful at an academic level at first because AEC lacks of it, the industry application would have to be checked. GORE was already used in industry through Enterprise Architecture to represent the strategic level of a system. Thus, its contribution to AEC was appropriate for this kind of missing information in AEC.

5   Research method

The adopted research method comes from Design Science and is structured around two axes. The first axis concerns the research activities in Design Science that are: build and evaluate. According to March & Smith [21], build refers to the construction of the artefact and aims at demonstrating that such an artefact can be constructed. Evaluate refers to the development of criteria and the assessment of the artefact's performance against those criteria. The second axis concerns the research outputs. Four kinds of outputs are defined and used to describe the PhD methodology: constructs, models, method, and instantiation.

The first output consists of building the constructs. A literature review of the concepts used in RE and other engineering domain (e.g. mechanical engineering) would allow me to gather the right concepts to define the building system. The main structure of these concepts comes from the RE as demonstrated in the section 3. This conceptual structure aims to be close to the clients' vocabulary and will be completed with the vocabulary of the architects and programmers on a technical level. This technical level will be built from vocabulary used in current AEC research (e.g. in Building Information Modelling [22]) or practice (e.g. ISO Standard [23]) to be used and useful.

In order to verify this structure, a sample of case studies from past construction projects (first case study retained: a multimedia library built in 2008 in Brittany, France) and interviews with AEC experts (i.e. architects, programmers, and contractors) would be led. For each artefact including constructs, a first case study would be implemented by the PhD student. Other case studies would be outsourced to trainees or students in a study project. This assessment aims to ensure feasibility, relevance, and completeness of the constructs and to test if the proposed solution is useful and usable by professionals of the AEC domain. A first measurement would be to compare the original case studies Design Brief with the corresponding implemented constructs. Difference between information provided in the original document and potential information generated following the constructs would illustrate the value added by it. A second measurement would be to ask other people (e.g. architectural students, programmers, architects) to use and complete such a structure on their own and gather their feedback on it. Last measurement would consist in processing the information implemented in a database to produce a different Design Brief that would satisfy at least the same requirements. Of course, all retained concepts from RE would not be necessarily useful in AEC even if their interest was highlighted. There was also no absolute certainty that architects and programmers would change their way of working using this structure yet. Thus, further investigations were required.

RE provided leads to answer AEC issues presented in this dissertation but it was not enough. On top of these elements related to RE, other engineering domain concepts would be studied to complete the building's constructs. These concepts would come from a state of the art on conceptual design methods in manufacturing and industrial engineering. First literature review has already revealed that conceptual design methods from other engineering domain were applied to architectural programming [15]. None of them started from a level of requirements this high including traceability to low level specifications. In this PhD study, a building would be specified from an advanced description of the organisation that would live inside.

6 Progress and Future Works

This PhD work has begun in February 2011. The first months were mainly dedicated to the understanding of the AEC domain in one part (mainly the architectural programming), and a more synthetic understanding and analysis of the conceptual phase (i.e. requirements phase) in other engineering domains (e.g. Manufacturing Engineering, Software Engineering). From a global analysis of this literature review, the research method was chose and detailed.

In the current state of the work, a conceptual view was designed and it integrated part of the operational level of constructs without the timing structure (i.e. definition of the deliverables' content). A database would be designed in May quickly followed by the evaluation process of constructs with experts and case study. The objective is to end the constructs part before June 2012.

The building of models would start right after the database, around beginning of June 2012, and their evaluation would be planned for around September 2012. The end of the year 2012 would be dedicated to the IT proof of concept, tool specification and to the method. The tasks plan for the year 2013 would hopefully include the instantiation of the constructs, models, and method on a case study or two. It would mostly involve the writing up of the PhD thesis.

References

1. Abdul-Kadir, M., Price, A.: Conceptual phase of construction projects. International Journal of Project Management. 13, 387-393 (1995).
2. Yu, A.T.W., Shen, G.Q.P., Kelly, J., Hunter, K.: Comparative Study of the Variables in Construction Project Briefing / Architectural Programming. Journal of Construction Engineering and Management. 122-138 (2008).
3. Yahya, I.A., Rahman, H.A., Berawi, M.A., Karim, S.B.A., Yee, K.L.: Value Management in analyzing project brief. Quantity Surveying International Conference. p. 12. , Kuala Lumpur, Malaysia (2007).
4. Kamara, J.M., Anumba, C.J., Evbuomwan, N.F.O.: Client requirements processing in construction: a new approach using QFD. Journal of architectural engineering. 5, 8 (1999).
5. Berry, D.M.: More requirements engineering adventures with building contractors. Requirements Engineering. 8, 142–146 (2003).
6. Zowghi, D., Coulin, C.R.: 02 Requirements Elicitation : A Survey of Techniques, Approaches, and Tools. Engineering and Managing Software Requirements. pp. 19-46. Springer Verlag (2005).
7. Kamara, J.M., Anumba, C.J., Evbuomwan, N.F.O.: Capturing client requirements in construction projects. Thomas Telford Ltd (2002).
8. Certu: Pour des bâtiments durables - Guide et outils de programmation. (2010).
9. MIQCP: Programmation des constructions publiques. Le Moniteur, Mission Interministérielle pour la Qualité des Constructions Publiques (2001).

10. Hickey, A.M., Davis, A.M.: An Ontological Approach to Requirements Elicitation Technique Selection. In: Sharda, R. and Voss, S. (eds.) Ontologies - A Handbook of Principles, Concepts and Applications in Information Systems. pp. 403–431. Springer (2007).
11. ISO/IEC: FDIS 9126-1 Software Engineering - Product quality - Part 1_Quality model. (1999).
12. Standish Group: CHAOS report. (1995).
13. Boehm, B.W., Papaccio, P.N.: Understanding and controlling software costs. IEEE Transactions on Software Engineering. 14, 1462-1477 (1988).
14. Arkley, P., Riddle, S.: Overcoming the Traceability Benefit Problem. , Newcastle (2005).
15. Mauger, C., Schwartz, T., Dantan, J.-Y., Harbouche, L.: Improving users satisfaction by using requirements engineering approaches in the conceptual phase of construction projects: The elicitation process. 2010 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). pp. 310–314. IEEE (2010).
16. Lamsweerde, A.V.: Goal-Oriented Requirements Engineering - A Guided Tour. 5th IEEE International Symposium on Requirements Engineering RE'01. p. 14. , Toronto (2001).
17. Zave, P., Jackson, M.: Four dark corners of requirements engineering. ACM Transactions on Software Engineering and Methodology (TOSEM). 6, 1–30 (1997).
18. Berry, D.M.: Software and House Requirements Engineering : Lessons Learned in Combating Requirements Creep. Requirements Engineering. 242-244 (2000).
19. Wieringa, R.J.: Software requirements engineering: the need for systems engineering and literacy. Requirements Engineering. 6, 132–134 (2001).
20. Glinz, M.: A Glossary of requirements engineering terminology. International Requirements Engineering Board (2011).
21. March, S.T., Smith, G.F.: Design and natural science research on information technology. Decision Support Systems. 15, 251-266 (1995).
22. Kiviniemi, A.: Requirements management interface to building product models. Stanford University Stanford, CA, USA (2005).
23. ISO: Normes de performance dans le bâtiment - Liste de contrôle consultative - Contenu d'un programme de conception dans l'industrie du bâtiment. (1994).

The Severity of Undetected Ambiguity in Software Engineering Requirements

Cristina Ribeiro

Department of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, N2L 3G1, Canada, cribeiro@uwaterloo.ca

Abstract. This paper discusses the potential negative effects of ambiguous software engineering requirements documents, and discusses a method of quantifying the severity of ambiguities in at least three software projects. The author theorizes that most ambiguities are naturally resolved through the process of software engineering and that the types of ambiguities that are likely to persist are those that people are not aware of. These types of ambiguities are most likely to suffer from subconscious disambiguation. When an ambiguity is disambiguated correctly, there are no negative effects, but when an ambiguity is disambiguated incorrectly, problems may arise. By identification of ambiguities that are most likely to suffer from subconscious disambiguation, we will calculate the costs associated with them, for each of the three software projects.

Keywords: Ambiguity, Costs, Effects of Ambiguities, Natural Language, Requirements Specifications, Requirements Engineering, Subconscious Disambiguation.

1 Introduction

Requirements engineering (RE)'s primary goal is to capture in a requirements specification (RS) all of the requirements that the client, users, and all stakeholders believe to be imperative in the computer-based system (CBS) being developed. Ambiguity in a RS could cause programmers to implement the CBS incorrectly, from the client's viewpoint, resulting in major code re-writes, leading to delays in delivery and introducing even more defects.

Almost all RSs are written in natural language (NL) [1]. Even when a RS is written with a formal language or UML diagrams, it still begins in NL [2]. NLs are inherently ambiguous. Therefore, in today's practice of RE, ambiguous RSs are pervasive [12].

Project failure has often been attributed to ambiguities in RS documents. For example, Gause lists too much unrecognized disambiguation in RSs as one of the five most important sources of requirements failure [3]. This attribution claim has not been conclusively proven, empirically, but this claim has fueled research in methods and tools for removing ambiguities in RSs, e.g., for writing less ambiguous requirements [5-9], for detecting ambiguities with tools [10-16] or manually [17-19], and for using restricted languages to write RSs [20-22].

Time is the single most important factor in software engineering. Since fixing a bug late in a software development life cycle is expensive [4], it is important to try to find all wrongly disambiguated ambiguities early. Doing so requires finding ambiguities early. The earlier an ambiguous requirement is found the less expensive it is to fix it. The least expensive time to find ambiguous requirements is in the analysis phase, before any development begins. The paradox is that finding ambiguities is expensive as it involves multiple, time-consuming, and focused inspections.

1.1 Success Criteria

There is little empirical evidence to support the claim that ambiguous requirements cause project failure. In a study on the effects of ambiguity on project success, de Bruijn analyzed one failed project and was unable to pinpoint the reason for the failure. He concluded there were a variety of factors that played a role in the failure of the project.

The difficulty with failed projects is that the reason for their failure is already known. If the reason for failure was not ambiguity then the probability of also ambiguity being the problem, is low. Due to this difficulty and the complexity of analyzing a failed project for ambiguities, the deliberate approach in this research is to quantify multiple *successful* projects to learn about their severity.

It is important to note that projects can be only seemingly successful, in that they may contain hidden defects that have not yet been found. I am trying to demonstrate that it is worth the effort required to search for really tough ambiguities. The search will try to find tough ambiguities that get overlooked and lead to a false sense of project success. I am searching for something serious that was overlooked and is the result of an undetected ambiguity that suffered from SD.

The success criteria for this research includes quantifying in time and costs associated to the severity identified for all ambiguities verified to have suffered from SD. Using this data we can answer the following question:
Which is more expensive:
1. letting subconsciously disambiguated ambiguities cause their damage and then be fixed late, or
2. doing enough focused inspections on the RS to find the ambiguities early before development starts?

1.2 Significance

Knowing the answer to the question stated above, enables requirements analysts and project managers to allocate their resources efficiently. They can decide whether it benefits them to use any of the ambiguity detection, avoidance, and or elimination tools that exist.

This research will determine how many SD ambiguities there are in this set of successful projects. The quantification of the severity of ambiguities suffering from SD and the costs associated with them is currently unknown.

There are also significant scientific contributions as a result of this research. The research and scientific community will benefit through the furthering of the body of knowledge, through the use of the research findings, the collected empirical data and the method used to collect this data. Similarly conducted experiments are highly valuable to industry and the software engineering community. This enables researchers to further this knowledge and it opens up many new related research hypotheses, which require examination.

2   Research Problem

A problem with all of these tools and methods to avoid or detect ambiguities is that we don't actually know if any of them are worth the effort. It would be valuable to know exactly what the benefits are and if ambiguity is in fact costly. De Bruijn [23] tried to work on solving this problem.

His overall goal was to determine the effect of ambiguity in a project's RS on the project's success. He analyzed a RS document for one failed CBS development project. With his analysis he attempted to answer the research questions [23]:

1. How many requirement statements are ambiguous?
2. How many problems were caused by ambiguous requirements?

De Bruijn's analysis found that only one defect in the CBS was caused by ambiguous requirements. The independent test team and the third party development team had been able to work through all the other ambiguities and successfully implement the specified CBS.

De Bruijn's conclusion was that for the RS and CBS he examined, the ambiguities that remained were not critical and had nothing to do with the failure. So he questioned whether focusing on ambiguities with special inspections and tools during RS is cost effective. Perhaps the normal conversation among stakeholders is sufficient to find the RS ambiguities that would cause defects in the developed CBS. As did de Bruijn, I wonder if ambiguity has an effect on project success and how much effort should be devoted to the avoidance and or detection of ambiguous requirements.

My research questions are refinements of de Bruijn's. The research questions are:

1. How many ambiguous requirements suffer from SD?
2. How many problems are caused by these ambiguities?
3. What is the severity of these problems?
4. Is the effort required to identify these ambiguities worth it?

In my research, a different approach is taken to deal with the size explosion problem he encountered. This approach is outlined in detail in the next section, the proposed solution.

3 Proposed Solution

3.1 Dealing with the Size Explosion Problem

De Bruijn's strategy included considering all kinds of ambiguities for one failed project. He ended up taking a random sample of the ambiguities, due to the large number of ambiguities he found and the inordinate amount of time it would take for him to review them all. This random sampling could have missed a lot of ambiguities, some of which may have caused their damage through the implementation of them in the code. Ambiguous requirements that cause a lot of damage are referred to as "show stoppers". Show stoppers may be too infrequent for them to be caught adequately with a random sampling. The fact that de Bruijn didn't catch any doesn't mean they don't exist. If you sample only 10% of the requirements, then you have a 90% chance of missing a show stopper, because show stoppers are so infrequent.

De Bruijn found that very few ambiguities affected development, because normal conversations during requirements analysis takes care of the ones people know about. The result was the ambiguities he found had no effect for the very reason that they were resolved through the natural process of software engineering. For example, a requirements engineer may ask a client "What do you mean by that?" to get more information. Also, everyone is aware of the problem of coordinating "and"s and "or"s, because the problem is encountered in everyday life, as in restaurant menus. Customers pick the interpretation that gives them the most, and the restaurant picks the interpretation that gives the customers the least.

I am using a different approach to deal with the size explosion, one that I believe is likely to produce better results. My strategy is to focus on the ambiguities likely to have been missed by stakeholders during requirements analysis and RS production, ambiguities that are likely to remain after analysis of the requirements by the stakeholders. These ambiguities should be more likely to cause expensive problems, requiring fixing late in the development.

3.2 Subconscious Disambiguation

The ambiguities that are likely to be missed by the stakeholders are those that suffer subconscious disambiguation (SD). SD of an ambiguity occurs when the reader or hearer of an ambiguous statement is not aware of the ambiguity and believes that his or her first understanding of the statement is the only possible understanding. In some cases, SD leads to the correct understanding relative to the meaning the writer intended, and sometimes it does not. When SD of an ambiguity in a RS leads to an understanding different from the client's original intent, the final CBS delivered to the client may be incorrect.

4   Research Method

De Bruijn analyzed one RS document for a *failed* project. I will be analyzing at least three RS documents, each belonging to a *successful* project. The RS documents will be analyzed to identify any subconsciously disambiguated ambiguities and to determine the effects they had on their projects.

The rest of this section describes how data will be collected, how the requirements will be inspected, and how severity of any ambiguities identified is measured. Using this procedure, I expect to be able to answer the research questions posed in Section 2.

We have a partnership with a major company, hereinafter referred to as "X". X has supplied high quality RS documents for three major CBSs that have been successfully implemented.

The number of documents available from X is probably not enough to have statistically significant results. However, each document is quite lengthy, averaging 90 pages. The amount of time required to carefully review these documents manually, is significant. Additional similar research will increase the total data sample and will strengthen the results found in this study.

I will review each RS in its entirety, searching for only ambiguities that are likely to have suffered SD. Once I have my list of ambiguities likely to have suffered SD, I will set up a meeting with X's chief requirements specification analyst. We will examine the histories of the developments of the CBSs for signs that the ambiguities I found causes development problems. The severity of these development problems will be estimated. These severities will be used to answer the research questions. Note that it is possible that I might find a problem that the X developers never even thought about and that does not show up in any project history. The company should be interested in hearing about such a problem, because it points to a potential flaw that has not yet been discovered. If such a problem proves to be severe, the argument that searching for ambiguities during RE is important is strengthened.

I am in the midst of the review of the first RS. Even when considering only ambiguities likely to have suffered SD, I have identified more than 300 ambiguities in the first RS document. It is unrealistic to expect the X analyst to devote the time necessary to determining the effect of each ambiguity. As a result, I am currently trying to determine a way to rank these ambiguities by likelihood of being show stoppers, with the intention of presenting to the X analyst the most likely show stoppers first.

The ranking process consists of three steps. The first step is grouping many similar ambiguities together minimizes the number of ambiguities to review. The second step is analyzing and documenting the possible alternate meanings and potential foreseeable effects for each ambiguity recorded. The third step is ranking the ambiguities in order of their potential severity level. At this point in the research, I can only rank these ambiguities in terms of what would likely be a show stopper. Once I confirm which ambiguities actually caused damage and assess how much damage they cause, I can then create a better categorization for the ranking of ambiguities.

5 Preliminary Results

At this point the requirements document for one of the three successful projects, has been reviewed. The RS document was reviewed in its entirety, for ambiguities likely to suffer from SD. Through the analysis process other ambiguities, not likely to have suffered from SD, were identified. In total there were 300 ambiguities, in the document, 41 pages in length. A subset of 24 ambiguities, were identified to have likely suffered from SD.

6 Research Progress

I am in the early stages of my research. Currently, only one of three RS documents has been reviewed. All the ambiguities likely to have suffered from SD have been identified and ranked by likelihood of being show stoppers. I am preparing to present these findings to the X analyst. The review of the first document has been going slower than I expected because I have had to refine my research method as I learned new things during the review. The other two reviews should go faster. I estimate that it will take another 8 to 10 months to gather the data necessary to answer the research questions.

References

1. Mich, L., Franch, M., and Novi Inverardi, P.: Market research for requirements analysis using linguistic tools. In: Requirements Engineering, 9, (1 & 2), pp. 40-56 (in No. 1) 151 (in No. 2). (2004)
2. Berry, D.M.: Ambiguity in Natural Language Requirements Documents. In: Lecture Notes in Computer Science, pp. 1-7. (2008)
3. Gause, D.C.: User DRIVEN Design – The Luxuray that has Become a Necessity. In: A Workshop in Full Life-Cycle Requirements Management, ICRE 2000 Tutorial T7, Schaumberg, IL, (2000)
4. Boehm, B.W.: Software engineering economics. Prentice-Hall (1981)
5. Goetz, R., Rupp, C.: Regelwerk natuerlichsprachliche methode. Technical report, Sophist (1999), http://www.sophist.de
6. Berry, D.M., Kamsties, E.: The syntactically dangerous *all* and plural in specifications. In: IEEE Software 22, 55–57 (2005)
7. Berry, D.M., Kamsties, E., Krieger, M.: From contract drafting to software specification: Linguistic sources of ambiguity. Technical report, University of Waterloo, Waterloo, ON, Canada (2003), http://se.uwaterloo.ca/ ~dberry/handbook/ambiguityHandbook.pdf
8. Kovitz, B.L.: Practical Software Requirements: A Manual of Content and Style. Manning, Greenwich, CT, USA (1998)
9. Dupré, L.: Bugs in Writing: A Guide to Debugging Your Prose, 2nd edn. Addison-Wesley, Reading (1998)
10. Osborne, M., MacNish, C.: Processing natural language software requirement specifications. In: Proceedings of the International Conference on Requirements Engineering (ICRE 1996), pp. 229–236 (1996)
11. Wilson, W.M., Rosenberg, L.H., Hyatt, L.E.: Automated analysis of requirement specifica-

tions. In: Proceedings of the Nineteenth International Conference on Software Engineering ICSE 1997, pp. 161–171. ACM Press, New York (1997)

12. Mich, L., Garigliano, R.: Ambiguity measures in requirement engineering. In: Feng, Y., Notkin, D., Gaudel, M. (eds.) Proceedings of International Conference on Software—Theory and Practice ICS 2000. Sixteenth IFIP World Computer Congress, pp. 39–48. Publishing House of Electronics Industry, Beijing (2000)

13. Kiyavitskaya, N., Zeni, N., Mich, L., Berry, D.M.: Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Requirements Engineering Journal 13, 207–240 (2008)

14. Berry, D.M., Bucchiarone, A., Gnesi, S., Lami, G., Trentanni, G.: A new quality model for natural language requirements specifications. In: Proceedings of the International Workshop on Requirements Engineering: Foundation of Software Quality, REFSQ 2006 (2006)

15. Tjong, S.F., Hartley, M., Berry, D.M.: Extended disambiguation rules for requirements specifications. In: Proceedings of Workshop in Requirements Engineering, WER (2007), http://wer.inf.pucrio.br/index.html

16. Tjong, S.F.: Avoiding Ambiguity in Requirements Specifications. PhD thesis, Faculty of Engineering & Computer Science, University of Nottingham, Malaysia Campus, Semenyih, Selangor Darul Ehsan, Malaysia (2008)

17. Kamsties, E., Berry, D.M., Paech, B.: Detecting ambiguities in requirements documents using inspections. In: Lawford, M., Parnas, D.L. (eds.) Proceedings of the First Workshop on Inspection in Software Engineering (WISE 2001), pp. 68–80 (2001)

18. Kamsties, E.: Surfacing Ambiguity in Natural Language Requirements. PhD thesis, Fachbereich Informatik, Universitaet Kaiserslautern, Kaiserslautern, Germany (2001); also Volume 5 of Ph.D. Theses in Experimental Software Engineering, Fraunhofer IRB Verlag, Stuttgart, Germany (2001)

19. Denger, C.: High quality requirements specifications for embedded systems through authoring rules and language patterns. Master's thesis, Fachbereich Informatik, Universitaet Kaiserslautern, Kaiserslautern, Germany (2002)

20. Comer, J.: An experimental natural-language processor for generating data type specifications. SIGPLAN Notices 18, 25–33 (1983)

21. Enomoto, H., Yonezaki, N., Saeki, M., Chiba, K., Takizuka, T., Yokoi, T.: Natural language based software development system tell. In: O'Shea, T. (ed.) Advances in Artificial Intelligence, ECAI 1984, pp. 721–731. Elsevier, Amsterdam (1984)

22. Fuchs, N., Schwertel, U., Schwitter, R.: Attempto controlled english (ACE) language manual version 3.0. Technical Report No. 99.03, Institut fuer Informatik der Universitaet Zuerich, Zuerich, Switzerland (1999)

23. de Bruijn, F., and Dekkers, H.: Ambiguity in Natural Language Software Requirements: A Case Study, In: Requirements Engineering: Foundation for Software Quality, pp. 233-247. (2010)

Previously published ICB - Research Reports

2012

No 51 (May)

Frank, Ulrich: "Specialisation in Business Process Modelling: Motivation, Approaches and Limitations"

No 50 (February)

Adelsberger, Heimo; Drechsler, Andreas; Herzig, Eric; Michaelis, Alexander; Schulz, Philipp; Schütz, Stefan; Ulrich, Udo: "Qualitative und quantitative Analyse von SOA-Studien – Eine Metastudie zu serviceorientierten Architekturen"

2011

No 49 (December 2011)

Frank, Ulrich: "MEMO Organisation Modelling Language (OrgML): Focus on Business Processes"

No 48 (December 2011)

Frank, Ulrich: "MEMO Organisation Modelling Language (OrgML): Focus on Organizational Structure"

No 47 (December 2011)

Frank, Ulrich: "MEMO Organisation Modelling Language (OrgML): Requirements and Core Diagram Types"

No 46 (December 2011)

Frank, Ulrich: "Multi-Perspective Enterprise Modelling: Background and Terminological Foundation"

No 45 (November 2011)

Frank, Ulrich; Strecker, Stefan; Heise, David; Kattenstroth, Heiko; Schauer, Carola: "Leitfaden zur Erstellung wissenschaftlicher Arbeiten in der Wirtschaftsinformatik"

No 44 (September 2010)

Berenbach, Brian; Daneva, Maya; Dörr, Jörg; Frickler, Samuel; Gervasi, Vincenzo; Glinz, Martin; Herrmann, 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 2011)

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. P. Chamoni MIS and Management Science / Operations Research	Information Systems and Operations Research, Business Intelligence, Data Warehousing
Prof. Dr. F.-D. Dorloff Procurement, Logistics and Information Management	E-Business, E-Procurement, E-Government
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. B. Müller-Clostermann Systems Modelling	Performance Evaluation of Computer and Communication Systems, Modelling and Simulation
Prof. Dr. K. Pohl Software Systems Engineering	Requirements Engineering, Software Quality Assurance, Software-Architectures, Evaluation of COTS/Open Source-Components
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