

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Barbara Paech Colette Rolland (Eds.)

Requirements Engineering: Foundation for Software Quality

14th International Working Conference, REFSQ 2008
Montpellier, France, June 16-17, 2008
Proceedings

Volume Editors

Barbara Paech
Universität Heidelberg
Im Neuenheimer Feld 326, 69120 Heidelberg
Germany
E-mail: paech@informatik.uni-heidelberg.de

Colette Rolland
Université Paris 1, Panthéon Sorbonne
90 Rue de Tolbiac, 75013 Paris
France
E-mail: Colette.Rolland@univ-paris1.fr

Library of Congress Control Number: 2008928136

CR Subject Classification (1998): D.2.1, D.2, F.3, K.6.1, K.6.3

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743
ISBN-10 3-540-69060-3 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-69060-3 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media
springer.com

© Springer-Verlag Berlin Heidelberg 2008
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12281411 06/3180 5 4 3 2 1 0

Preface

The 14th Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2008) will be held in the beautiful city of Montpellier, France, June 16–17, 2008.

The main topic was fitness of requirements engineering. Our economic productivity and well-being in every-day life strongly hinges on information technology, and thus software quality. In most systems, quality is determined through the development process. With the spread of service-oriented and autonomic systems, software quality is continuously negotiated and adapted at run-time. Requirements Engineering sets the stage for quality, both at development- and run-time. In spite of the constant emergence of new technologies and development paradigms, basic issues such as effective communication between stakeholders or correctness, consistency and completeness of large requirements documents are the dominant issues in industry.

Seventeen papers written by authors from 13 different countries address these topics, with particular focus on elicitation, innovative systems and empirical studies, as well as industrial experiences and maturing research. Within these themes, the work presented spans a wide range of application domains such as public health, aeronautics and the automotive industry. It also involves a variety of requirements engineering techniques, from the most established (such as use cases or feature models) to the most innovative (such as search-based software engineering or negotiation constellations).

As in the previous years, the proceedings serve as a record of REFSQ 2008, but also present an excellent snapshot of the state of the art of research and practice. As such, we believe it is of interest to the whole requirements engineering community, from students embarking on their PhD to experienced practitioners, researchers and teachers interested in emerging knowledge, techniques and methods.

March 2008

Barbara Paech
Colette Rolland
Patrick Heymans
Anne Persson

Organization

REFSQ is run by an Organizing Committee of two Program Co-chairs and two Organizational Co-chairs appointed by a permanent advisory board. REFSQ 2008 was co-located with CAiSE 2008.

Advisory Board

Eric Dubois (CRP Henri Tudor, Luxembourg)
Andreas L. Opdahl (University of Bergen,
Norway)
Klaus Pohl (University of Duisburg-Essen,
Germany)

Organizing Committee

Program Co-chairs	Colette Rolland (Université Paris 1 Panthéon Sorbonne, France) Barbara Paech (University of Heidelberg, Germany)
Organizational Co-chairs	Patrick Heymans (University of Namur, Belgium) Anne Persson (University of Skövde, Sweden)

Program Committee

I. Alexander	V. Gervasi	E. Kamsties
T. Alspaugh	J.-P. Giraudin	J. Krogstie
A. Aurum	M. Glinz	R. Laeau
C. Baron	M. Goedicke	S. Lauesen
D.M. Berry	J. Gordijn	J. Leite
J. Börstler	T. Gorschek	M. Lemoine
S. Brinkkemper	A. Herrmann	L. Liu
P.-J. Charrel	P. Heymans	P. Loucopoulos
L. Chung	A. Hickey	N. Madhavji
A. Davis	J. Huang	N. Maiden
E. Dubois	M. Jarke	R. Matulevicius
C. Ebert	M. Jirotko	D.M. Moody
G. Fanmuy	S. Jones	J. Natt och Dag
A. Finkelstein	N. Juristo	C. Ncube

B. Nuseibeh
A. Olive
A.L. Opdahl
A. Persson
K. Pohl
C. Potts
N. Prakash
J. Ralyte
B. Ramesh

L. Rapanotti
B. Regnell
M. Rossi
A. Russo
C. Salinesi
K. Sandahl
P. Sawyer
K. Schneider
A. Silva

G. Sindre
I. Sommerville
J. Stirna
R. Wieringa
C. Wohlin
E. Yu
D. Zowghi

Table of Contents

REFSQ 2008

REFSQ 2008 International Working Conference on Requirements Engineering: Foundation for Software Quality.....	1
<i>Barbara Paech and Colette Rolland</i>	
Process Improvement in Requirements Management: A Method Engineering Approach	6
<i>Sjaak Brinkkemper, Inge van de Weerd, Motoshi Saeki, and Johan Versendaal</i>	
Enhancing Elicitation Technique Selection Process in a Cooperative Distributed Environment.....	23
<i>Hakim Bendjenna, Nacereddine Zarour, and Pierre-Jean Charrel</i>	
Negotiation Constellations – Method Selection Framework for Requirements Negotiation	37
<i>Samuel Fricker and Paul Grünbacher</i>	
DESCRY: A Method for Evaluating Decision-Supporting Capabilities of Requirements Engineering Tools	52
<i>Beatrice Alenljung and Anne Persson</i>	
Inventing Requirements: Experiences with an Airport Operations System	58
<i>Neil Maiden, Cornelius Ncube, and James Lockerbie</i>	
A Stakeholder Model for Interorganizational Information Systems	73
<i>Luciana C. Ballejos, Silvio M. Gonnet, and Jorge M. Montagna</i>	
Search Based Requirements Optimisation: Existing Work and Challenges	88
<i>Yuanyuan Zhang, Anthony Finkelstein, and Mark Harman</i>	
Connecting Feature Models and AUTOSAR: An Approach Supporting Requirements Engineering in Automotive Industries	95
<i>Wolfram Webers, Christer Thörn, and Kurt Sandkuhl</i>	
Using a Creativity Workshop to Generate Requirements for an Event Database Application.....	109
<i>Claudia Schlosser, Sara Jones, and Neil Maiden</i>	
Can We Beat the Complexity of Very Large-Scale Requirements Engineering?	123
<i>Björn Regnell, Richard Berntsson Svensson, and Krzysztof Wnuk</i>	

Macro-level Traceability Via Media Transformations	129
<i>Orlena C.Z. Gotel and Stephen J. Morris</i>	
Towards Simulation-Based Quality Requirements Elicitation: A Position Paper	135
<i>Roland Kaschek, Christian Kop, Vladimir A. Shekhovtsov, and Heinrich C. Mayr</i>	
Classifying Assumptions Made during Requirements Verification of Embedded Systems	141
<i>Jelena Marinčić, Angelika Mader, and Roel Wieringa</i>	
Integrating Portfolio Management and Simulation Concepts in the ERP Project Estimation Practice	147
<i>Maya Daneva</i>	
Can Patterns Improve i* Modeling? Two Exploratory Studies	153
<i>Markus Strohmaier, Jennifer Horkoff, Eric Yu, Jorge Aranda, and Steve Easterbrook</i>	
Discovering Web Services to Improve Requirements Specifications: Does It Help?	168
<i>Konstantinos Zachos, Neil Maiden, and Rhydian Howells-Morris</i>	
Mobile Discovery of Requirements for Context-Aware Systems	183
<i>Norbert Seyff, Florian Graf, Paul Grünbacher, and Neil Maiden</i>	
When to Adapt? Identification of Problem Domains for Adaptive Systems	198
<i>Kristopher Welsh and Pete Sawyer</i>	
Author Index	205

REFSQ'08 International Working Conference on Requirements Engineering: Foundation for Software Quality

Barbara Paech¹ and Colette Rolland²

¹ University of Heidelberg, Im Neuenheimer Feld 326, D-61920 Heidelberg
paech@informatik.uni-heidelberg.de

² Centre de Recherche en Informatique, Université Paris1 Panthéon Sorbonne,
90 Rue de Tolbiac, 75013 Paris, France
Colette.Rolland@univ-paris1.fr

Abstract. The 14th Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'08) will be held in the beautiful city of Montpellier, France on the 16th and 17th June 2008. This introduction gives an overview of the conference and its program.

1 Introduction

Requirements Engineering – Foundation for Software Quality (REFSQ) is the unique requirements engineering (RE) event having an explicit mission to promote the many roles of quality in RE. After 11 successful years as a workshop it evolved in 2006 into a working conference with attendance opened beyond the accepted paper authors. Last year for the first time a keynote was invited and the proceedings were published for the first time as Springer Lecture Notes in Computer Science. This year the overall format is essentially retained. Sjaak Brinkkemper will give the invited keynote. There will be parallel sessions, but based on the experience with 27 accepted papers last year, more time is given again this year for the individual paper presentations. This is important to keep the highly interactive and participatory nature of the conference. With the help of the largest program committee ever, 17 out of 50 submissions have been accepted representing again a healthily selective acceptance rate of 34%.

2 The Program

While REFSQ is open to all papers focusing on quality in RE, it has a specific topic each year. This year the topic was fitness of RE. Our economic productivity and well-being in every-day life strongly hinges on information technology, and thus software quality. In most systems, quality is determined through the development process. With the spread of service-oriented and autonomic systems, software quality is continuously negotiated and adapted at run-time. Requirements Engineering sets the stage for quality, both at development- and run-time. In spite of the constant

emergence of new technologies and development paradigms, basic issues such as effective communication between stakeholders or correctness, consistency and completeness of large requirements documents are the dominant issues in industry. Thus, RE has to accommodate for many different processes and products adapted to these diverse situations.

The seventeen papers presented provide an excellent snapshot of the state of the art of research and practice in RE. They were grouped into the following sessions:

2.1 Fitness

The papers in the first session of the conference focus on the specific topic, in particular the assessment of fitness in terms of RE techniques and RE tool selection. All approaches give experience-based guidance for this assessment. In *Elicitation Technique Selection Process in Cooperative Distributed Environment: Why is it Different?* Hakim Bendjenna¹, Nacereddine Zarour and Pierre-Jean Charrel extend existing work on elicitation technique selection to handle stakeholder conflicts and to take into account more stakeholder characteristics like the language. Samuel Fricker and Paul Grünbacher, in *Negotiation Constellations – Method Selection Framework for Requirements Negotiation*, address for the first time the problem of negotiation technique selection. The idea is to base the selection on negotiation constellations which capture the negotiation characteristics of the software organization and of the negotiating parties, and differentiate negotiation tactics and methods. In the position paper *DESCRY: An Evaluation Method for Assessing Decision-supporting Capabilities of RE Tools*, Beatrice Alenljung and Anne Persson propose 9 criteria for evaluating RE tools wrt. decision support.

2.2 Requirements Elicitation

As in the previous year, a number of papers focused on elicitation. In *Inventing Requirements: Experiences with an Airport Operation System* Neil Maiden, Cornelius Ncube and James Lockerbie report on the combination of creativity techniques and use cases. For the first time they could evaluate in detail the impact of the creativity techniques. Luciana Ballejos, Silvio Gonnet and Jorge Montagna describe *A Stakeholder Model for Inter-organizational Information Systems*. This model allows capturing quantitatively different roles and their interests in and influences on the system. It is illustrated in the context of a public health care system. In a project involving many stakeholders, it is also important to quantitatively explore the benefits and drawbacks of requirements. In the position paper *Search Based Requirements Optimisation: Existing Work & Challenges* Yuanyuan Zhang, Anthony Finkelstein, and Mark Harman present a vision for solving this problem with search-based methods. These methods offer several advantages like robustness and sensitivity analysis, but also induce some challenges such as scalability and the definition of the fitness function.

2.3 Industrial Experience of RE

Several papers in the conference report industrial experience. The papers in this session present detailed insight into the challenges of real life RE. In *Connecting Feature*

Models and AUTOSAR: An Approach Supporting Requirements Engineering in Automotive Industries, Wolfram Webers, Christer Thörn, and Kurt Sandkuhl discuss challenges for suppliers in the automotive domain. While AUTOSAR provides a standard for the exchange of requirements between OEM and supplier, the suppliers still face the problem of relating requirements documents of different customers to the assets of their product line. The paper presents a case study to bridge this gap based on feature models. In *Using a Creativity Workshop to Generate Requirements for an Event Database Application*, Claudia Schlosser, Sara Jones and Neil Maiden present lessons learned by performing a creativity workshop. On the one hand a detailed description of the workshop is given, and on the other hand the outcome in terms of number and quality of the generated requirements is analyzed. The last paper of this session, *Can We Beat the Complexity of Very Large-Scale Requirements Engineering?*, pinpoints a notorious problem of RE in industry: the size of the documents. The authors, Björn Regnell, Richard Svensson and Krzysztof Wnuk, define different scales and then focus on very large scale RE concerning over 10.000 requirements with strong interdependencies. Based on their experiences the authors propose sustainable requirements architectures, effective requirements abstraction and emergent quality prediction as most promising future RE research topics.

2.4 Innovative Systems

This session collects the papers dealing with the question how RE needs to be adapted to innovative systems. Web services are the focus of *Discovering Web Services to Improve Requirements Specifications: Does It Help?* by Konstantinos Zachos, Neil Maiden and Rhydian Howells-Morris. The paper investigates the usefulness of requirements-based tools supporting the specification of queries and the search in service registries. In *In-situ Discovery of Requirements for Mobile and Context-aware Systems: How Scenario-based Approaches Can Help?*, the authors, Norbert Seyff, Florian Graf, Paul Grünbacher and Neil Maiden, explore how RE tools based on ubiquitous technology can support the RE for ubiquitous systems. The lessons learned from the usage of such tools for the validation of scenarios give rise to a number of requirements for ubiquitous RE tools, such as on-site usage, unobtrusive use and detection of context change. Furthermore, research challenges are derived. Context dependencies are also the focus of *When to Adapt? Identification of Problem Domains for Adaptive Systems* by Kristopher Welsh and Pete Sawyer. In their position paper the authors argue that dynamically adaptive systems are needed especially in case of context-dependent variation in the acceptable trade-offs between non-functional requirements.

2.5 Maturing Research

The position papers in this session present first ideas on innovative RE techniques. The authors of the first paper, Orlena Gotel and Stephen Morris, discuss *Macro-Level Traceability via Media Transformations*. The idea is to retain the transformation steps between different media such as transcription of an interview or structuring of informal text into use cases. This eases the traceability between up-stream RE and down-stream RE. The second position paper, *Towards Simulation-Based Quality*

Requirements Elicitation: A Position Paper, by Roland Kaschek, Christian Kop, Vladimir Shekhovtsov and Heinrich Mayr proposes the simulation of business processes to support elicitation of quality requirements. One key idea is to simulate the environment of the system. A formal model of the system and the environment is also important in the third paper, *Classifying Assumptions Made During Requirements Verification of Embedded Systems*, by Jelena Marincic, Angelika Mader and Roel Wieringa. One major challenge is to make sure that the formal models correspond with the intended systems. This paper argues that the confidence in this correspondence is enhanced by retaining the assumptions made, and furthermore that a classification of these assumptions can help to guide the modeling process.

2.6 Empirical Studies

The two papers in this session do not focus on the presentation of new techniques, but rather on their empirical evaluation. Maya Daneva evaluates data from two industrial sites in *Integrating Portfolio Management and Monte Carlo Simulation Concepts in ERP Project Estimation Practice: a Case Study*. A case study and an experiment are the basis of *Can Patterns improve i* Modeling? Two Exploratory Studies* by Markus Strohmaier, Jennifer Horkoff, Eric Yu, Jorge Aranda, and Steve Easterbrook. In this context patterns are reusable i* models. The collected data does not support the expected reduction of effort or complexity, but shows improved model coverage.

3 Concluding Remarks

Table 1 gives a breakdown of the national affiliations of the accepted papers' authors. It shows how many papers have one or more authors affiliated with a particular country, not the number of authors from each country. Some papers are co-authored by pan-national teams, so the sum of the numbers in table 1 exceeds the number of papers accepted. This year, the strongest contribution came from UK, while last year's winner, Germany, is at the lower end.

Table 1. REFSQ'08 Author Affiliations by Country

Country	Papers (co-)authored
Algeria	1
Argentina	1
Austria	4
Canada	1
France	1
Germany	1
New Zealand	1
Sweden	3
Switzerland	1
The Netherlands	2
The Ukraine	1
United Kingdom	7
United States	1

We are writing this before REFSQ'08 has taken place. All readers who are interested in an account of the discussions that took place during the conference and the subsequent conclusions should consult the post-conference summary which we intend to publish in the ACM SIGSOFT Software Engineering Notes.

Acknowledgements

REFSQ'08 is very much a collaborative effort involving many people. First of all, we would like to thank Eric Dubois, Andreas Opdahl and Klaus Pohl who served on the REFSQ Advisory Board, and Patrick Heymans and Anne Persson, this year's organisational co-chairs.

We would also like to thank the members of the program committee who acted as anonymous reviewers and provided valuable feedback to the authors:

Ian Alexander, Thomas Alspaugh, Aybüke Aurum, Claude Baron, Daniel M. Berry, Jürgen Börstler, Sjaak Brinkkemper, Pierre-Jean Charrel, Lawrence Chung, Alan Davis, Éric Dubois, Christof Ebert, Gauthier Fanmuy, Anthony Finkelstein, Vincenzo Gervasi, Jean-Pierre Giraudin, Martin Glinz, Michael Goedicke, Jaap Gordijn, Tony Gorschek, Andrea Herrmann, Patrick Heymans, Ann Hickey, Jane Huang, Matthias Jarke, Marina Jirotko, Sara Jones, Natalia Juristo, Erik Kamsties, John Krogstie, Régine Laleau, Søren Lauesen, Julio Leite, Michel Lemoine, Lin Liu, Peri Loucopoulos, Nazim H. Madhavji, Neil Maiden, Raimundas Matulevičius, Daniel M. Moody, Johan Natt och Dag, Cornelius Ncube, Bashar Nuseibeh, Antoni Olive, Andreas Opdahl, Anne Persson, Klaus Pohl, Colin Potts, Naveen Prakash, Jolita Ralytė, Bala Ramesh, Lucia Rapanotti, Björn Regnell, Matti Rossi, Alessandra Russo, Camille Salinesi, Kristian Sandahl, Peter Sawyer, Kurt Schneider, Andres Silva, Guttorm Sindre, Ian Sommerville, Janis Stirna, Roel Wieringa, Claes Wohlin, Eric Yu and Didar Zowghi.

Finally, we are very grateful to Andreas Classen for web, logo and document design, to Germain Saval for his prompt maintenance of the website, to Doris Keidel-Müller for her support in reviewing the layout of the papers, and to Willi Springer for all his help and hard work during the whole of the REFSQ'08 life-cycle – even on weekends.

Process Improvement in Requirements Management: A Method Engineering Approach

Sjaak Brinkkemper¹, Inge van de Weerd¹, Motoshi Saeki², and Johan Versendaal¹

¹ Department of Information and Computing Sciences

Utrecht University, Utrecht, The Netherlands

{s.brinkkemper, i.vandeweerd, j.versendaal}@cs.uu.nl

² Department of Computer Science

Tokyo Institute of Technology, Tokyo, Japan

saeki@cs.titech.ac.jp

Abstract. Method Engineering and Requirements Engineering are two research fields that can benefit from another. To increase process maturity in systems development, we propose an approach for incremental method evolution that combines capability-based and problem-based methods. With this method, we can assemble new methods, based on the process need of an organization. We show how this approach can be implemented using Computer Aided Method Engineering (CAME) technology. In addition, we demonstrate the utility of the Product Software Knowledge Infrastructure by showing an example of the insertion of cost-value prioritization as a method increment in software product management. This shows how isolated innovations in the Requirements Engineering domain can be embedded in software development practices.

Keywords: method engineering, requirements engineering, software process improvement, incremental method evolution, root cause analysis, computer aided method engineering, CAME.

1 Method Engineering and Requirements Engineering

The research areas of Method Engineering and Requirements Engineering share a common interest, as they both aim at promoting process improvements in software and systems developments. Method Engineering works from the perspective of generic method descriptions, usually called meta-models and possibly supported by tooling, that allow for the roll-out of uniform high-quality methods in the perspective of full means for situational adaptation of the method to the circumstances at hand.

Requirements Engineering research focuses on all techniques for the proper description and handling of the specifications of a systems development process, or as Nuseibeh and Easterbrook formulate more formally, the process of discovering the software system's purpose, by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation [1]. In the scientific work of requirements engineering we see all kinds of innovative approaches being proposed related to the elicitation, modeling and analysis, communicating, validating and evolution of requirements.

This paper aims at establishing a cross-fertilization of the two perspectives by showing how requirements engineering techniques can be embedded into a systems development method supported by method engineering principles. We demonstrate this by inserting a cost-value requirements prioritization technique, developed by Karlsson and Ryan [2], into the requirements management methods of a product software company [3].

1.1 Methods for Product Software Development

Product software is a worldwide industry, yet this domain has not been subject of much scientific research. The last years, this is changing however. There have been several studies on all product software, focusing on product software as a research domain [4], product development [5] [6], management of software products [3] [7], requirements management [8], release planning [9] [10], product line engineering [11] [12], product delivery [13], and so on.

Xu and Brinkkemper [4] summarize a number of specific characteristics of developing product software. An important difference is, for example, that the production costs do not depend on the number of copies sold. Therefore, product software companies that are selling millions of copies can have up to 99% gross profit margins for its product sales [4]. On the other hand, the majority of the product-development project are late or over budget. Also, the requirements of the entire market must be held into account. This means that a software product should be developed so that it can run on different hardware and software platforms. All these characteristics make product software development a highly complex business, in which process failures have a huge impact on performance.

Furthermore, as is depicted in Figure 1, the success of a software product in the market has consequences for the internal functioning of the company. From a start-up creating a first release product by a relatively simple process, the growing company is

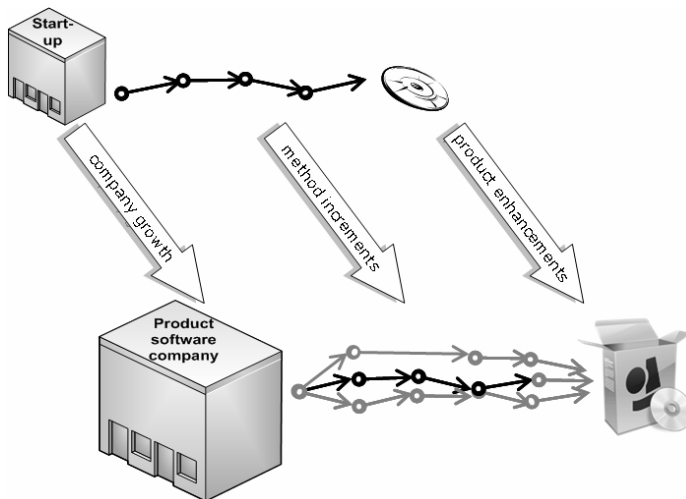


Fig. 1. Incremental method evolution in a product software company

shipping subsequent releases and product enhancements by utilizing a product development approach that should be incrementally adapted to the changing conditions.

Several software process improvement approaches have been proposed to improve software development processes [14] [15]. These approaches are usually capability-based, i.e. based on the current capabilities of a company an advice is given which entails the implementation of capabilities on a higher maturity level. However, the increments in these approaches are often too large and general, instead of local and situational. For example, SEI has done a survey among 1,804 organizations, which indicates that the median time, to move from one CMM level to another, ranges from thirteen to twenty-four months [16].

In this research, we want to extend the capability-based process improvement with root-cause analysis, in order to give a more accurate analysis of the actual problem. We implement our approach in the Product Software Knowledge Infrastructure (PSKI) [17] [18], which, when fully materialized, can help to increase the maturity of a company's processes. For scoping reasons we limit our research to the software product management domain.

1.2 Research Approach

This research project is carried out following the design research methodology for performing research in information systems as described by [19] and [20]. Research in design science is done through the processes of *building* and *evaluating* artifacts [19] [20]. According to Hevner et al. [19], the fundamental questions in design-science research are: "*What utility does the new artifact provide?*" and "*What demonstrates that utility?*" In addition, they provide seven guidelines on performing design-science that have been followed during this research. The first guideline Hevner et al. propose is that "design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation". The artifact in this research is the Product Software Knowledge Infrastructure (PSKI), or to be more specific, the functional architecture of the PSKI. The second guideline is *problem relevance*, which Hevner et al. describe as "the objective of design-science research is to develop technology-based solutions to important and relevant business problems". The business problem lies in the fact that product software market is growing and that there is a need for methodical support, in order to increase the maturity of product software organizations. By developing the PSKI, we offer a technology-based solution to this problem. The other guidelines comprise: design evaluation, research contributions, research rigor, design as a search process, and communication of research. The page length of this paper limits us to describing each guideline in detail.

In earlier work [17], we described our vision on this issue and introduced the PSKI, our main new artifact. Subsequently, in [18], we identified and formalized general method increments that were found in an exploratory case study. In addition, we formalized common process needs, by developing a root-cause map for software product management and by identifying the root causes and process alternatives that are related to them. Finally, a first prototype of a method base for software product

management is developed¹, based on the reference framework for software product management [3].

In this research we want to elaborate on the process improvement approach that will be implemented in the PSKI and its functional architecture of the PSKI. We evaluate this by using scenarios to demonstrate its utility. In Section 2, we will describe the realization of the PSKI, by elaborating on the requirements and functional architecture. Section 3 explains the technical realization of integrating the PSKI with a CAME tool. In section 4, we will give a scenario of a method increment, advised and assembled by the PSKI. Then, in Section 5, we give an overview of related literature. Finally, in section 6, we will describe the conclusions and further research.

2 Realization of the Product Software Knowledge Infrastructure

In this section we will first describe the rationale of the software improvement approach we use. Then we describe the functional architecture of the PSKI and show a typical scenario.

2.1 A Combined Process Improvement Approach

We propose the distinction between two types of process improvement approaches: the capability based and problem-based approach. The capability-based approach is based on the assumption that a company's capabilities should grow in maturity in order to increase performance. By assessing the organization's current capabilities, the maturity level can be determined and recommendations of implementing capabilities on a higher maturity level can be made. Examples of capability-based approaches are CMM [14] and SPICE [15]. Secondly, the problem-based approach uses the mechanism of solving the underlying problems, or root causes, that cause a certain process to under perform. An example of a problem-based approach is RCA, which has been applied to process improvement and incident prevention in software and non-software industries; see for example [21].

In literature, some critique exists on capability-based approaches. For example, a capability-based approach is can be encountered as too superficial for small companies [22]. In addition, these kinds of process improvement approaches are often difficult to implement. In [23], it was found that CMMi, is often not adopted by organizations because the following reasons: the organization was small; the services were too costly, and the organization had no time to implement the process improvements. From our experience, we also found that capability-based approaches often are too superficial for the specific nature of product software companies. More over, we do not want to force companies to a company-wide process improvement program. On the other hand, following a complete problem-based approach would be too inefficient, due to the extensive analysis process that needs to be done. Therefore, we propose the *combined process improvement approach*, in which we complement the capability-based approach with problem-based aspects. When comparing this approach to the existing capability-based approaches such as SPICE and CMM, we envision the following advantages: 1) the maturity levels can be determined per process, which makes it possible to implement very small process improvements;

¹ <http://www.softwareproductmanagement.org/>

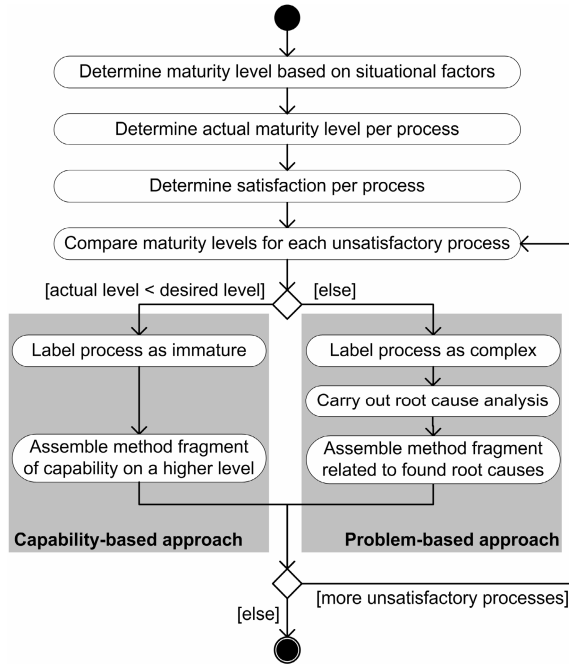


Fig. 2. Process improvement approach

2) the capability-based approach is extended with a problem-based approach to be able to determine the more complex problems that underlay a unsatisfactory process. In Figure 2, we illustrate this approach.

The process starts with determining the maturity level that the company *should have*, based on the situational factors of the company. For example, a company with 500 employees should be on a higher maturity level than a company with six employees. Secondly, the actual maturity levels per process are retrieved by performing a capability assessment. By inventorying which capabilities are mastered per process, the maturity level can be calculated. In addition, the user is asked whether the result of the concerned process is satisfactory or unsatisfactory. For each unsatisfactory process then, the maturity level as it *should be* (based on situational factors) and the actual maturity level are compared. If the actual maturity level is lower than the maturity level based on situational factors, then the process is labeled as immature and a process improvement is necessary. The process improvement is carried out by assembling a method fragment related to the capability on a higher maturity level. If the actual level is equal to or higher than the desired level, then the process is labeled as complex, and a root-cause analysis is carried out to find the underlying problems. The process improvement is carried out by assembling a method fragment, related to the found root causes.

2.2 Functional Architecture

Starting from [17], the following components in the PSKI (see Figure 3) can be identified: a *web-based interface* to communicate with the user; an *assessment base*,

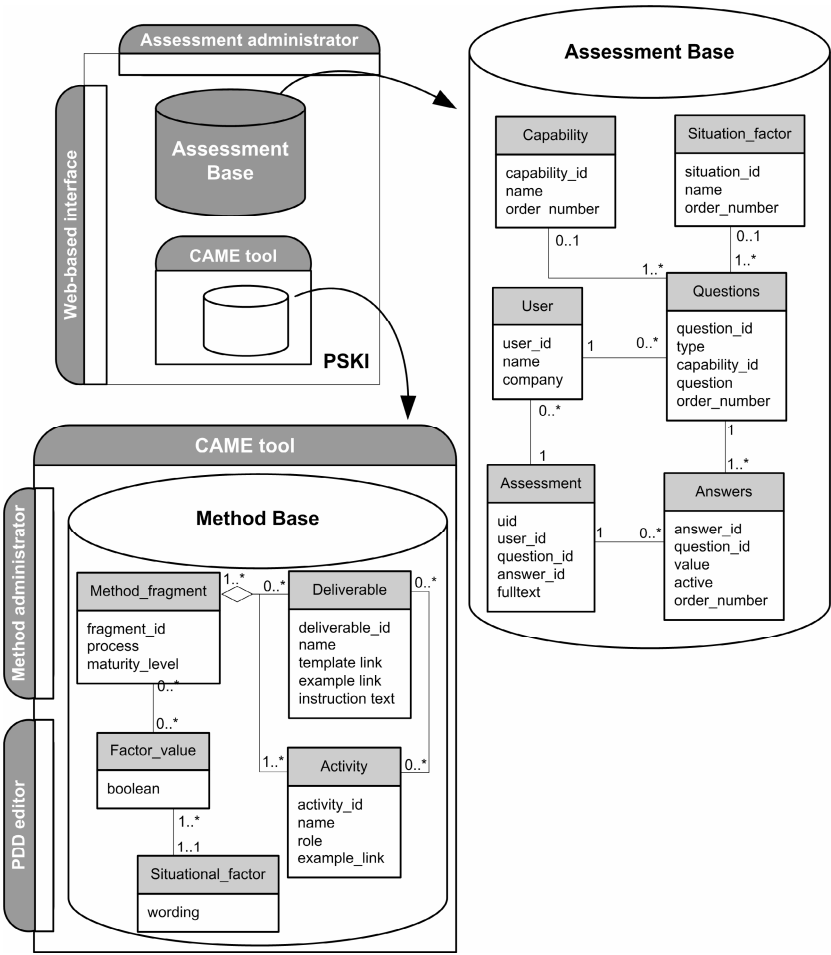


Fig. 3. Functional architecture of the PSKI

to store the assessment questions and answers; the *assessment administrator*, which can be used to add questions to the *assessment base*; and the *CAME tool* in which the method fragments are stored.

The main components of the PSKI are the assessment base and the CAME tool. In the assessment base, the PSKI stores assessment questions, answers, situational factors and capabilities. We distinguish two types of questions: *situation questions* that identify which situational factor apply to a company or product line and *capability questions* that assess which capabilities a company possesses. The second component is the CAME tool, which consists of a *method base*, in which method fragments their information are stored; a *PDD editor*, with which the method engineer can define the meta-modeling language that is used for administrating the methods; and the *method administrator* that is used by the method engineer to add methods to the method base. The method fragments that are stored in the method base consist of

activities and deliverables. Each method fragment is labeled with a capability level and linked to zero or more values of situational factors.

2.3 Illustrative Example: ERPComp

To illustrate the utility of the PSKI, we use a running example, which concerns an organization that develops ERP systems (ERPComp). ERPComp is 3 years old and currently has 50 employees. The user in this case is the product manager of the organization, who uses the PSKI because his organization has several problems: a) the releases are often not delivered in time, and b) the stakeholders are not satisfied with the implemented requirements.

Figure 4 illustrates the current requirements management process of ERPComp in PDD notation [24]. The diagram shows a snapshot of the method at a certain time n , say method increment #0. It covers the requirements management activity of a company, and has two sub activities: Gather requirements, resulting in a REQUIREMENT and Write release definition, resulting in a RELEASE DEFINITION, which are both carried out by the product manager.

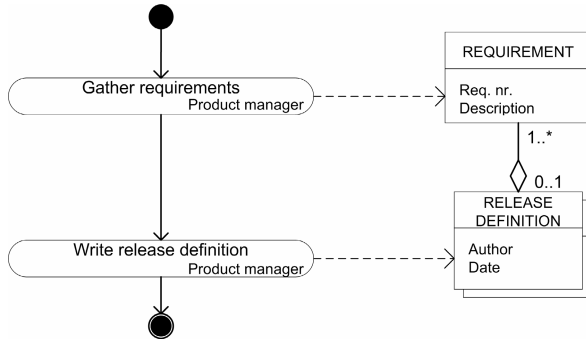


Fig. 4. Snapshot of increment #0

2.4 A Typical Scenario in the PSKI

In Figure 5, we show again the functional architecture of the PSKI enriched with the process that is followed when interacting with the PSKI. We will elaborate on this process by using the ERPComp example. Note that in this case, the capability-based approach is followed, as indicated in Figure 5. By following the solid arrows, the activities concerning the problem-based approach (*Present root cause map* and *Store root causes*) are skipped.

The scenario depicted in Figure 5 describes a sequence of the following activities:

PSKI: Present situational questions

PSKI presents a form with a predefined set of situational assessment questions.

User: Answer situational questions

The product manager answers situational questions:

1. What is the age of your organization (in years)?

☐ <1 ☒ 1-5 ☐ 5-10 ☐ >10

2. In which sector does your organization operate?

Large-sized enterprises ▼

3. What is the size of the development team?

☒ 1-4 ☐ 5-9 ☐ 10-20 ☐ >20

...

8. What is the number of product lines?

☒ 1 ☐ 2-4 ☐ 5-8 ☐ >9

9. Which platform is used to develop your product on?

.NET ▼

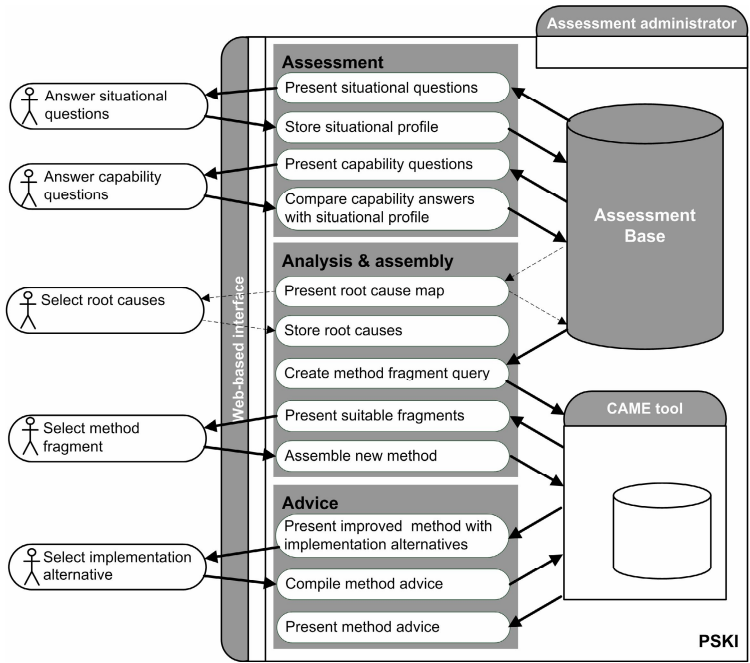


Fig. 5. Capability-based PSKI scenario

PSKI: Store situational profile

PSKI stores the answers to the situational questions as a situational user profile in the assessment base. Also, based on this profile, the desired maturity level is obtained. In this case, based on the age of the organization (3 years) and the sector in which the organization operates, the PSKI determines the maturity level at 4 (of 12, see Section 3.2).

PSKI: Present capability questions

Based on the answers to the situational questions, PSKI selects a subset from the capability questions, namely those questions that have the same type as indicated in by the user in his situational answers. This means that only questions that are applicable for large-sized organizations, with multiple products, developed on a .NET platform, are selected. Examples of these questions are:

1. Does your organization perform requirements prioritization per release?	<input type="radio"/> yes	<input checked="" type="radio"/> no
2. Is there a prioritized requirements list?	<input type="radio"/> yes	<input checked="" type="radio"/> no
3. Is the prioritized requirements list properly available for other stakeholders?	<input type="radio"/> yes	<input checked="" type="radio"/> no
4. Is there a Product Manager responsible for the requirements prioritization per release?	<input type="radio"/> yes	<input checked="" type="radio"/> no

User: Answer capability questions

The product manager answers the capability questions.

PSKI: Compare capability answers with situational profile

PSKI stores the answers to the capability assessment questions as a capability profile in the assessment base. When comparing the capability profile with the desired maturity level, the PSKI finds the following:

Process	Right capabilities in place?	Result satisfactory?
Requirements gathering	Yes	Yes
Requirements validation	No	No
Requirements prioritization	No	No

In case the product manager would have found the requirements gathering process unsatisfactory, although the process was at the right maturity level, the PSKI would present the root cause map of this process. However, due to limited space, we will not elaborate on such an example. The remaining steps are therefore:

PSKI: Create method fragment query

PSKI creates method fragment query, which retrieves those method fragments that are linked to the capabilities that should be implemented, in this case the level-3 capabilities of the *Requirements validation* and *Requirements prioritization* processes.

PSKI: Present suitable answers

PSKI displays all matching method fragments. Below, we depict an example of two method fragments for the Requirements prioritization process.

1.	Requirements prioritization via a stakeholder voting round <i>Description:</i> The product manager schedules a meeting in which each stakeholder gives his top x of requirements that need to be implemented in the next release. The requirements with the most votes will be implemented. <i>Roles:</i> Product manager, involved stakeholders <i>Deliverables:</i> REQUIREMENTS LIST with prioritized REQUIREMENTS
2.	Requirements prioritization via the cost-value approach <i>Description:</i> In the cost-value approach, the relative costs and relative values of each requirement are estimated. Then, they are plotted on a cost-value diagram, which shows which requirements will generate the highest value and the lowest costs. Based on this diagram, the product manager prioritizes the requirements. <i>Roles:</i> Product manager, product group, customers, software engineer <i>Deliverables:</i> COST-VALUE DIAGRAM, prioritized REQUIREMENTS

User: Select method fragments

The product manager selects method fragments that are perceived as useful.

PSKI: Assemble new method

PSKI assembles the selected method fragment into the existing method fragments of the company.

PSKI: Present method with implementation alternatives

PSKI presents method accompanied by a number of different implementation alternatives.

User: Select implementation alternative

The product manager selects suitable implementation alternative

PSKI: Compile method advice

PSKI compiles method advice is compiled, according to the selected implementation alternative. In case the user has selected root causes, an advice is added on how to solve these.

PSKI: Present method advice

PSKI presents the method advice to the product manager.

3 Method Improvement Based on Situational Capability Matching

In this section, we elaborate on the retrieving process of method fragments from the method base. Instead of building the method base ourselves, we use an existing tool, namely MetaEdit+. MetaEdit+ is an integrated modeling and meta-modeling environment for domain-specific languages [25] [26]. In MetaEdit+, we have realized our PDD notation as a meta-model. Now, it is possible to create, store and manipulate method fragments as PDDs. A screenshot of MetaEdit+ can be found at the end of this paper, in Figure 10.

3.1 Method Fragment Structure

A method fragment consists of a process fragment and a deliverable fragment. Method fragments can contain multiple activities and multiple deliverables. Also, constructs like branches, joining and forking of activities and aggregated deliverables can be modeled, as shown in Figure 8 and 9 and described in [24]. The structure of a generic method fragment is depicted in Figure 6.

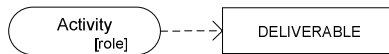


Fig. 6. Generic method fragment structure

The name of an activity in a method fragment is a composition of one or more verbs, possibly an adjective and a noun, e.g. *Prioritize [verb] requirements [noun]*. Furthermore, an activity is carried out by a role, e.g. *Product Manager [role]*.

The structure of method fragments is used in the generation of capability questions for the capability assessment. We distinguish two types of capability questions: standard questions, which can be generated from the stored activities, deliverables and

capabilities; and comprehensive questions, which are especially useful for assessing capabilities at a higher level. Based on the activities, deliverables and capabilities, we can derive the capability assessment questions. Each capability is related to three basic assessment questions, namely:

- 1. *Does your organization perform the [capability]?*
- 2. *Is there a [deliverable]?*
- 3. *Is the [deliverable] properly available for other stakeholders?*
- 4. *Is there a [role] responsible for the [capability]?*

In section 2.3, three capability assessment questions were listed. These were the assessment questions for capability A: Requirements prioritization per release, namely:

- 1. *Does your organization perform requirements the prioritization per release?*
- 2. *Is there a prioritized requirements list?*
- 3. *Is the prioritized requirements list properly available for other stakeholders?*
- 4. *Is there a Product Manager responsible for the requirements prioritization per release?*

In ERPComp, the product manager answers ‘no’ to all questions, since there is no requirements prioritization process in place.

3.2 Maturity Matrix for Software Product Management

To assess the state of the SPM function in an organization, we developed the SPM maturity matrix. This maturity matrix is inspired by on the DYA architecture maturity model [27] and the Test Process Improvement model [28]. We distinguish 16 SPM processes in the maturity matrix that originate from the reference framework for SPM [3] and 11 maturity levels. The number of maturity levels is determined by the implementation dependencies of the capabilities. In Table 1, we show an excerpt of the matrix, covering three processes. Each process has its own path to maturity, indicated by the letters A, B, C and D. Every letter represents a capability, which we define as the demonstrable ability and capacity to perform a certain process at a certain level. The position of the letters shows the preferred order in which the capabilities need to be implemented to reach a certain maturity level. 10 is the lowest maturity level and 12 is the highest maturity level. Suppose that a company should be on maturity level 4, based on its situational factors. This means that for Requirements prioritization, capabilities A and B should be implemented; for Requirements validation, capability A should be implemented; and for Requirements gathering, capabilities A and B should be implemented.

Table 1. Excerpt of the maturity matrix for Software Product Management

Process	Maturity level	1	2	3	4	5	6	7	8	9	10	11	12
Requirements prioritization				A		B		C			D		
Requirements validation					A		B		C			D	
Requirements gathering			A			B		C		D			

In ERPComp, the desired maturity level that is deducted from the situational user profile is level 3. We will elaborate on two processes in the SPM maturity matrix, namely requirements prioritization and requirements validation. Please note that although the capability structure of the requirements prioritization and requirements organizing processes are the same, this may vary in other processes.

In the requirements prioritization process we distinguish four capabilities:

- A. Requirements prioritization per release
- B. Requirements prioritization as an ongoing process
- C. Requirements prioritization as an ongoing process, over multiple product lines
- D. Requirements prioritization as a chain-wide process

Currently, no prioritization process is in place. Looking at the matrix, we see that level-3 companies should have capability A (Requirements prioritization per release) implemented.

For the requirements validation process, also four capabilities are distinguished:

- A. Requirements validation per release
- B. Requirements validation as an ongoing, automated process,
- C. Requirements validation as an ongoing process, over multiple products
- D. Requirements validation as a chain-wide process

Currently, there is no validation at all. A level-3 organization should master capability A: Requirements validation per release.

4 Method Increment Example

In this section, we illustrate a process improvement by a capability-based method increment. The snapshot of increment #0, that we showed in Figure 4, is created in MetaEdit+. This means that not only visual information is stored, but also extra information, depending on the variables that we added to the different concepts.

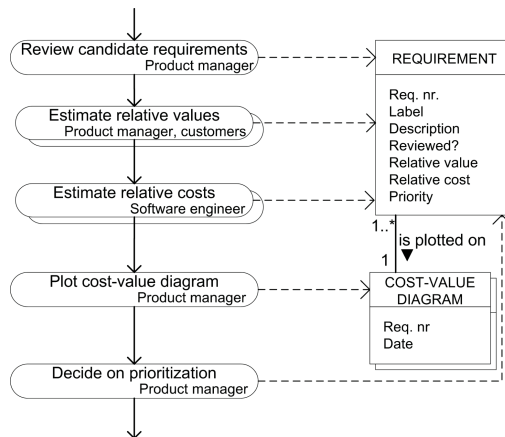


Fig. 7. Method fragment linked to 'Requirements prioritization per release'

As described in section 3.1, the organization should implement two method increments. The first method increment concerns the capability ‘Requirements prioritization per release’. As described in section 2.4, two method fragments are related to this capability. In this case, the user chooses the method fragment ‘Requirements prioritization via the cost-value approach’, as is depicted in Figure 7. The cost-value approach is proposed in [2] and evaluated in [29] as a method for requirements prioritization in market-driven software product development.

In Figure 8, we illustrate the method fragment related to the capability ‘Requirements validation per release’.

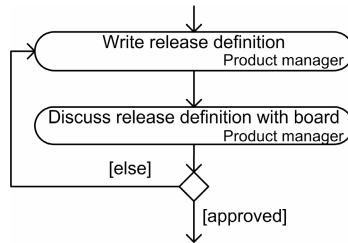


Fig. 8. Method fragment linked to 'Requirements validation per release'

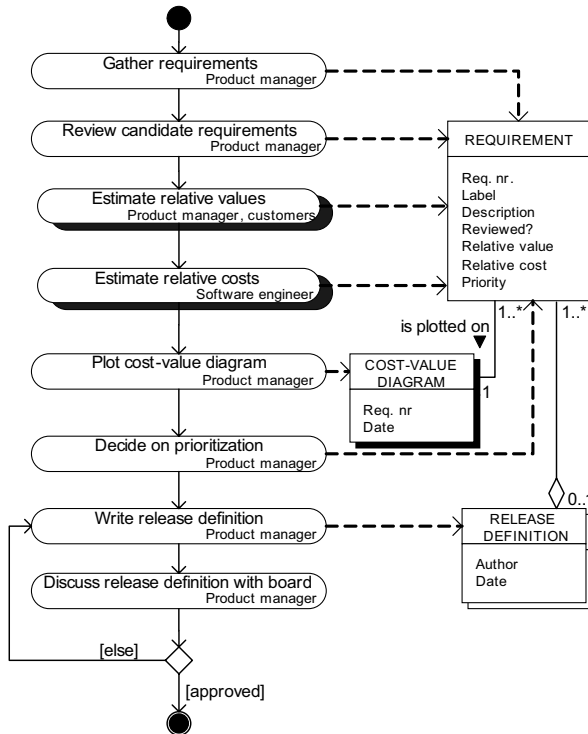


Fig. 9. Snapshot of increment #1

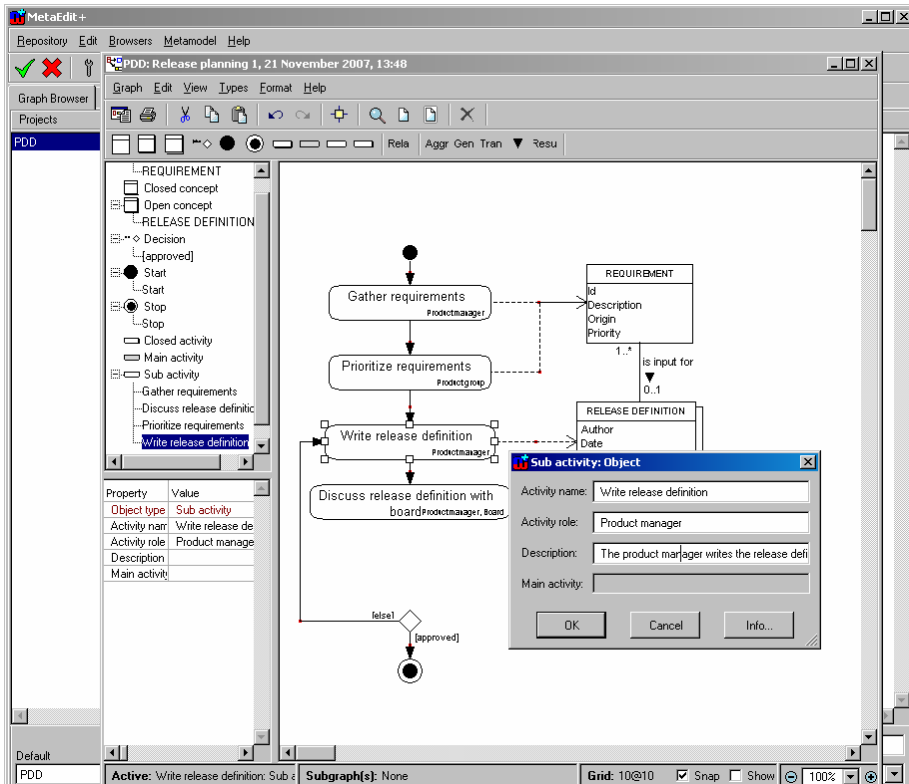


Fig. 10. Assembly of the improved method in MetaEdit+

The fragment does not have the standard form of activity – deliverable, but the activity results in a decision, indicated by a branch. The Product manager discusses the RELEASE DEFINITION with the board. If the board approves it, the release can be implemented. If not, the RELEASE DEFINITION has to be rewritten.

In Figure 9, we illustrate the snapshot of the improved method. It includes the method increments described in Figure 7 and 8. The roles of the activities are filled in based on the situational information that was provided during the situational assessment.

Finally, we want to show how we created the method fragments we presented in this paper. In Figure 10 we show a screenshot of MetaEdit+, in which a new method is modeled. Looking at the scenario we explained in Section 2.4, we can position this activity, although it is not automated yet, in the step ‘Assemble new method’. In the future, this activity will be automated.

5 Related Literature

In [30], it is stated that there is a scarcity of requirements engineering-related software process improvement initiatives in the literature. In addition, in [31] and [32], the authors state that existing software process improvement approaches leave a gap regarding requirements engineering. Therefore, they propose a practice-based

approach to requirements engineering process improvement. Also other studies have been done to process improvement in requirements engineering. For example, [33] describes a requirements engineering process improvement programme, based on lessons learned from the implementation of a requirements engineering approach for packaged software. The authors in [34] also point out that the requirements phase of software development is in need of further support. They propose the Requirements Capability Maturity Model (R-CMM) as a first step in the solution to this problem.

Product software developers use methods and techniques in all phases of the development process, which are often supported by different software tools. These software tools range from simple text editors to complex tools for generating code from design specifications. Not only techniques on a low level can be automated, but also methods, which focus more on the high-level activities and deliverables of a process, can be automated. In the nineties of the previous century, this led to a new research discipline, namely Method Engineering [35] [36] [37], which comprises the design, construction and adaptation of methods, techniques and tools for the development of ISs. Tools were being designed to support the method engineering process, which are called computer-aided method engineering (CAME) tools [26]. Many CAME tools have been developed in the last years, some for research purposes and some for commercial purposes. Their appliances vary from domain-specific modeling, to configuration management and situational method engineering.

6 Conclusions and Further Research

In this research, we proposed an approach for incremental method evolution that provides means by which innovative requirements engineering techniques can be inserted into systems development methods based on method engineering principles. This vision on process improvement combines a capability-based approach with problem-based aspects. We showed how this approach can be implemented in the PSKI by elaborating on the functional architecture. In addition, we explained the utility of the PSKI by giving an example of a method increment, i.e. cost-value requirements prioritization in software product management. This generic approach defines structure and relations of capabilities and method fragments, and generates capability questions automatically. Finally, we showed how method increments can be generated based on the situational and capability assessment answers.

We are currently working on the development of the PSKI and filling it with situational factors, capabilities and method fragments. In the future, we will use case studies to test the infrastructure at product software companies of different sizes and in different sectors, in order to test the mapping between situational factors, maturity capabilities and method fragments. We are confident that this paper shows how method engineering and requirements engineering research can benefit from each other.

References

1. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering, ICSE 2000, pp. 35–46. ACM, New York (2000)

2. Karlsson, J., Ryan, K.: A cost-value approach for prioritizing requirements. *IEEE Software* 14, 67–74 (1997)
3. van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L.: Towards a Reference Framework for Software Product Management. In: *Proceedings of the 14th IEEE International Requirements Engineering Conference*, pp. 319–322 (2006)
4. Xu, L., Brinkkemper, S.: Concepts of product software. *European Journal of Information Systems* 16, 531–541 (2007)
5. MacCormack, A.: Product-Development Practices that Work: How Internet Companies Build Software. *MIT Sloan Management Review* 42, 75–84 (2001)
6. Hietala, J., Kontio, J., Jokinen, J.P., Pyysiäinen, J.: Challenges of software product companies: results of a national survey in Finland. In: *Proceedings of the 10th IEEE International Symposium on Software Metrics*, pp. 232–243 (2004)
7. Ebert, C.: The impacts of software product management. *Journal of Systems and Software* 80, 850–861 (2007)
8. Regnell, B., Höst, M., Natoch Dag, J.N., Beremark, P., Hjelm, T.: An Industrial Case Study on Distributed Prioritisation in Market-Driven Requirements Engineering for Packaged Software. *Requirements Engineering* 6, 51–62 (2001)
9. van den Akker, M., Brinkkemper, S., van Diepen, G., Versendaal, J.: Flexible Release Planning Using Integer Linear Programming. In: *Proceedings of the 11th International Workshop on Requirements Engineering for Software Quality, Essener Informatik Beiträge, Band 10*, pp. 13–14 (2005)
10. Ruhe, G., Saliu, M.O.: The art and science of software release planning. *IEEE Software* 22, 47–53 (2005)
11. Pohl, K., Böckle, G., van der Linden, F.: *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, Heidelberg (2005)
12. Kang, K.C., Lee, J., Donohoe, P.: Feature-oriented product line engineering. *IEEE Software* 19, 58–65 (2002)
13. Jansen, S., Ballintijn, G., Brinkkemper, S., van Nieuwland, A.: Integrated development and maintenance for the release, delivery, deployment, and customization of product software: a case study in mass-market ERP software. *J. Softw. Maint. Evol.: Res. Pract.* 18, 133–151 (2006)
14. Paulk, M.C., Weber, C.V., Curtis, B., Chrissis, M.B.: *The capability maturity model: guidelines for improving the software process*. Addison-Wesley Longman Publishing, Boston (1995)
15. El Emam, K., Drouin, J.N., Melo, W.: *SPICE: the theory and practice of software process improvement and capability determination*. IEEE Computer Society Press, Los Alamitos (1998)
16. Process Maturity Profile Software CMM 2005 End-Year Update (2006) (retrieved March 22, 2008), <http://www.sei.cmu.edu/appraisal-program/profile/pdf/SW-CMM/2006marSwCMM.pdf>
17. van de Weerd, I., Versendaal, J., Brinkkemper, S.: A product software knowledge infrastructure for situational capability maturation: Vision and case studies in product management. In: *Proceedings of the Twelfth Working Conference on Requirements Engineering: Foundation for Software Quality, Luxembourg*, pp. 97–112 (2006)
18. van de Weerd, I., Brinkkemper, S., Versendaal, J.: Concepts for Incremental Method Evolution: Empirical Exploration and Validation in Requirements Management. In: Krogstie, J., Opdahl, A., Sindre, G. (eds.) *CAiSE 2007 and WES 2007*. LNCS, vol. 4495, pp. 469–484. Springer, Heidelberg (2007)
19. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. *MIS Quarterly* 28, 75–105 (2004)

20. March, S.T., Smith, G.F.: Design and natural science research on information technology. *Decision Support Systems* 15, 251–266 (1995)
21. Leszak, M., Perry, D.E., Stoll, D.: A case study in root cause defect analysis. In: *Proceedings of the 22nd international conference on Software engineering*, pp. 428–437 (2000)
22. Demirors, O., Demirors, E.: Software process improvement in a small organization: Difficulties and suggestions. In: Gruhn, V. (ed.) *EWSPT 1998*. LNCS, vol. 1487, pp. 1–12. Springer, Heidelberg (1998)
23. Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P., Murphy, R.: An exploratory study of why organizations do not adopt CMMI. *Journal of Systems and Software* 80(6), 883–895 (2007)
24. van de Weerd, I., Brinkkemper, S.: Meta-Modeling for Situational Analysis and Design Methods. *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications*, Information Science Reference, Hershey PA (2008)
25. Tolvanen, J.: MetaEdit+: integrated modeling and metamodeling environment for domain-specific languages. In: *OOPSLA 2006*, pp. 690–691. ACM, New York (2006)
26. Kelly, S., Lyytinen, K., Rossi, M.: MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. In: Constantopoulos, P., Vassiliou, Y., Mylopoulos, J. (eds.) *CAiSE 1996*. LNCS, vol. 1080, pp. 1–21. Springer, Heidelberg (1996)
27. Steenbergen, M., Brinkkemper, S., van den Berg, M.: An Instrument for the Development of the Enterprise Architecture Practice. In: *Proceedings of the 9th International Conference on Enterprise Information Systems*, pp. 14–22 (2007)
28. Koomen, T., Pol, M.: *Test Process Improvement: A Step-by-step Guide to Structured Testing*. Addison-Wesley Longman Publishing, Boston (1999)
29. Lehtola, L., Kauppinen, M.: Suitability of Requirements Prioritization Methods for Market-driven Software Product Development. *Software Process: Improvement and Practice* 11, 7–19 (2006)
30. Damian, D., Zowghi, D., Vaidyanathasamy, L., Pal, Y.: An Industrial Case Study of Immediate Benefits of Requirements Engineering Process Improvement at the Australian Center for Unisys Software. *Empirical Softw. Eng.* 9, 45–75 (2004)
31. Sawyer, P., Sommerville, I., Viller, S.: Requirements process improvement through the phased introduction of good practice. *Software Process Improvement and Practice* 3, 19–34 (1997)
32. Niazi, M.K.: Software Process Improvement: A Road to Success. In: *Proceedings of the Seventh Australian Workshop on Requirements Engineering*, pp. 125–139 (2002)
33. Regnell, B., Beremark, P., Eklundh, O.: A Market-driven Requirements Engineering Process: Results from an Industrial Process Improvement Programme. *Requirements Engineering* 3, 121–129 (1998)
34. Beecham, S., Hall, T.: Defining a Requirements Process Improvement Model. *Software Quality Journal* 13(3), 247–279 (2005)
35. Brinkkemper, S.: Method Engineering: Engineering of Information Systems Development Methods and Tools. *Inf. and Softw. Techn.* 38, 275–280 (1996)
36. Kumar, K., Welke, R.J.: Methodology Engineering: a proposal for situation-specific methodology construction. In: *Challenges and strategies for research in systems development*, pp. 257–269. John Wiley & Sons, Inc., New York (1992)
37. Rolland, C., Prakash, N.: A proposal for context-specific method engineering. In: *Proceedings of the IFIP TC8, WG8. 1/8.2 working conference on method engineering on Method engineering: Principles of method construction and tool support: principles of method construction and tool support*, pp. 191–208 (1996)

Enhancing Elicitation Technique Selection Process in a Cooperative Distributed Environment

Hakim Bendjenna^{1,2,3}, Nacereddine Zarour², and Pierre-Jean Charrel³

¹ University Center of Tebessa, Computer Science Department,
Tebessa, Algeria

bendjenna@univ-tebessa.dz

² LIRE Laboratory, Computer Science Department, University of Constantine
Constantine, Algeria

nasro-zarour@umc.edu.dz

³ Toulouse University and Institut de Recherche en Informatique de Toulouse
Toulouse, France

charrel@univ-tlse2.fr

Abstract. Requirements elicitation is a key stage in the successful designing of the computerized information system of a distributed organization. Few works have been focusing on how a requirements analyst selects one of the existing requirements elicitation techniques, notably in a distributed cooperative environment. However, the elicitation technique selection process creates significant communication, coordination, cultural and processes diversity challenges which impact the effectiveness of all the requirements engineering process and, further, product quality. This paper presents a decision making process that allows a requirements analyst to choose an elicitation technique in a cooperative distributed environment based on stakeholders' preferences, linguistic knowledge and priorities.

Keywords: requirements engineering, requirements elicitation, elicitation technique selection, distributed environment.

1 Introduction

Information systems and their embedded software are currently taking place in heterogeneous environments where people, information and working processes are distributed. Work is often cooperative and involves multiple actors who are the stakeholders of many kinds of requirements.

Requirements engineering is one of the early processes of the system development life cycle and it involves stakeholders in an iterative process of problem analysis, requirements elicitation, specification and validation [24], [26].

Requirements elicitation may be the most important area of requirements engineering and possibly of the entire software process [22]. It is generally accepted that errors produced at the requirements stage, if undetected until a later stage of software development, can be very costly [22], [25]. However, software engineers spend too little time in performing this important task [25].

Many issues related to the requirements elicitation process have been extensively analyzed in literature (see for instance [13] or [26] for survey). Most of these issues stress communication between stakeholders [23], which is critical during the requirements elicitation stage. Communication becomes even more difficult in a lot of present decentralized software projects whose stakeholders are distributed in several regions of the world. It generates new issues, like language and culture difference, time difference between sites [8].

Computer-Supported Cooperative Work (CSCW) and Cognitive Informatics are two research fields interested in communication issues. The former studies human behaviour within groups, and aims to focus on providing groupware tools, i.e. technologies which improve communication between stakeholders distributed along distant locations. The latter, as an interdisciplinary area, combines several disciplines, such as informatics, computing, software engineering, and cognitive sciences [7].

Our paper follows recent works related to these research areas. In particular, Hickey and Davis [17] presented a general model of the elicitation process which involves a selection phase among all requirements elicitation methodologies and techniques. Following, Aranda et al. [2] introduced some concepts from cognitive psychology, which help to evaluate the so-called "cognitive style" of the stakeholders, in order to propose a model which supports stakeholders' personal preferences in geographically distributed situations.

They extended their model [3] by adding features of distributed environments (e.g. time difference) and knowledge about stakeholders' preferences (e.g. knowledge level of a common language, stakeholders' characteristics). However, they underline in [2] and [3] that further work is needed to solve conflicts when stakeholders' preferences seem to be opposite.

At most one elicitation technique must be applied for all stakeholders in an iteration of the elicitation process. The primary research question investigated in the present paper is: *In a distributed cooperative environment, when the preferences of two or more stakeholders are at variance, (i.e. their comfort feeling with an elicitation technique is opposite) what is the appropriate elicitation technique that an analyst must choose? Or, in other words: How the analyst chooses an elicitation technique corresponding to the preferences of one stakeholder rather than the other(s)?*

We argue that the answer to this question must lean upon stakeholders' characteristics, linguistic knowledge, priorities, and finally analyst's preferences.

The remainder of this paper is organized as follows. Section 2 outlines requirements elicitation and elicitation technique selection in the requirements process. Section 3 describes the process model we propose to select the elicitation techniques. Two motivating examples are illustrated in Section 4. Section 5 summarizes related works on requirements in a cooperative distributed environment. The last section summarizes the results of this paper and outlines hints for future works.

2 Elicitation Technique Selection

Requirements elicitation is generally performed using an elicitation methodology which involves a series of techniques. These methodologies and techniques aim to

assist analysts in understanding requirements [21]. Though some analysts think that one methodology or one technique is sufficient to all situations, several works [21], [18] early showed these methodologies and techniques depend on the situation.

Several requirements elicitation techniques are used in distributed environment, such as question and answer methods, interviews, brainstorming, use cases, storyboards, prototyping, and questionnaire [20]. But, why an analyst decides using one technique or another? According to [18], there are four main reasons:

- it is the only one the analyst knows;
- it is the analyst's favourite technique, so she/he uses it for all situations;
- the analyst follows a methodology which advocates a particular technique;
- the analyst thinks (intuitively) the technique is the most effective in that situation.

This suggests that it is possible to improve the success of the product by selecting the elicitation technique, according to the current situation. These techniques are comparable and it is possible to improve the way techniques are selected [20].

According to this idea, Hickey and Davis [17] proposed a general model of an iterative elicitation process, based on the following principle: at iteration i , the elicitation technique is selected by means of the following selector function σ :

$$\sigma(R_i, S_i, \chi(T)) \rightarrow \{t\} \quad (1)$$

Where:

- $\chi(T)$ is a given set of characteristics of all elicitation techniques T ;
- $\{t\}$ is a set of elicitation techniques which can be applied in situation S_i (i.e. characteristics of problem and solution domain and the project) when the current state of requirements is R_i (i.e. the collection of requirements which have already been elicited);

$\chi(T)$ captures inherent features of elicitation techniques, such as their ability to help the analyst to: reduce ambiguity, resolve conflicts, converge towards a solution, raise new issues and so on. These characteristics are static and identical for all projects.

Selector function σ identifies the best possible match between the characteristics of the techniques and the current state of the requirements and situation. For example, if the requirements are unclear, techniques that reduce ambiguity may be helpful (e.g. the prototyping technique).

Then, another selector function π computes the best technique t_i as an intersection between the suggested techniques $\{t\}$ and the analyst's preferences P :

$$\pi(\{t\}, P) \rightarrow t_i \quad (2)$$

Aranda et al. [2], based on Hickey and Davis [17] work, proposed a model which links stakeholders' learning preferences to the requirements elicitation technique which would be the most suitable according to those preferences. They focus on instruments issued from the field of psychology called Learning Styles Models. The new Selector function π^* is not only based on the analyst's preferences but also on the preferences of all the stakeholders who participate in this iteration of the requirements elicitation process.

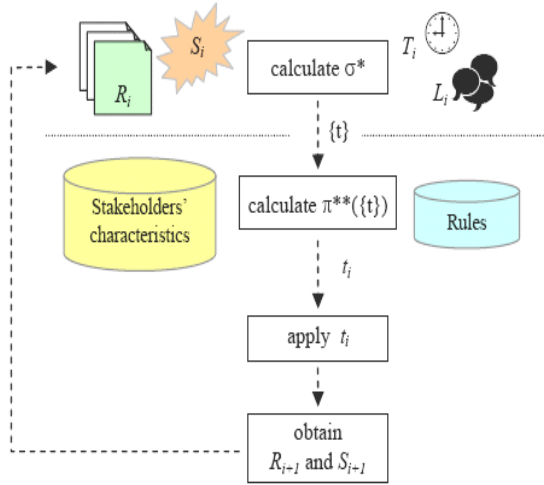


Fig. 1. The iterative process of requirements elicitation in a distributed environment [3]

Aranda et al. [3] also proposed a model based on an iterative process for elicitation techniques selection, in a distributed environment (cf. figure 1).

In this process σ^* is a selector function defined as follows:

$$\sigma^*(R_i, S_i, \chi(T), T_i, L_i) \rightarrow \{t\} \quad (3)$$

Where:

- $\chi(T)$ is a given set of characteristics of all elicitation techniques T ;
- T_i (time difference) indicates the level at which synchronous communication is possible between the sites which must interact. $T_i \in \{\text{no-overlap, little-overlap, half-overlap, much-overlap, full-overlap}\}$;
- L_i (knowledge of a common language) indicates the fluency of communication. $L_i \in \{\text{low, low-intermediate, intermediate, high-intermediate, high}\}$;
- $\{t\}$ is a set of elicitation techniques that can be applied in situation S_i when the current state of requirement is R_i according to restrictions T_i and L_i .

Then, another selector function π^{**} computes a suitable technique as follows:

$$\pi^{**}(\{t\}, (PS_1, ws_1), (PS_2, ws_2) \dots (PS_k, ws_k) \dots (PS_n, ws_n)) \rightarrow t_i \quad (4)$$

Where:

- PS_k is the set of techniques that fit the k -th stakeholder's preferences;
- ws_k is the weight of the preferences (i.e. how strong they are?);
- $t_i \in \{t\} \mid t_i \in PS_k \text{ and } ws_k = \max(ws_1, \dots, ws_n)$;

t_i is an appropriate elicitation technique for the current i -th situation and for the stakeholder whose personal preferences are the strongest.

The stakeholders' preferences techniques are obtained from stakeholders' characteristics, which result from Felder-Silverman's Learning Styles Model (LSM)

classification [15]. LSMs are used to analyse relationships between students and teachers. They classify students according to their behaviour when they learn a given task. Aranda et al. [3] considered an analogy between the roles in LSMs and the stakeholders in an elicitation process. For example, the requirements analyst learns from users, and vice versa. This model classifies people as follows (see [1], [2], [3] for details):

- *Sensing / Intuitive*. *Sensing* people have rather learning facts, while *Intuitive* people prefer discovering possibilities and relationships.
- *Visual / Verbal*. *Visual* people remember better what they see, while *Verbal* people prefer explanations.
- *Active / Reflective*. *Active* people understand and remember information when they do something, while *Reflective* people have rather thinking solely first.
- *Sequential / Global*. *Sequential* people understand easier when following a step by step procedure, while *Global* people try to get the rough features, to find connections and discover solutions in novel ways, even if they are not always able to explain them.

Every stakeholder is classified according to a multiple-choice test (available on the WWW¹). This test affects them a rank for each subcategory. A stakeholder may fit into several categories, depending on the circumstances: she/he may be sometimes reflective and sometimes active. Their preference for one category has a value the measure of which is strong, moderate, or mild. A stakeholder is classified as a member of a group, only if a strong preference can be measured for him.

One of the open issues is to solve conflicts when stakeholders' preferences seem to be opposite [3].

3 A Process Model for Elicitation Technique Selection in a Cooperative Distributed Environment

3.1 Motivations to Improve the Selection Process

According to us, the process model presented in [3] can be improved from the four following arguments [5]:

1. The model does not cope with the situation where stakeholders' preferences are opposite.
2. A stakeholder can be classified as a member of a group only if she/he has strong preferences, while the weight of preferences (ws_k) used in function π^{**} , can be strong, moderate or mild.
3. The requirements analyst is treated in function π^{**} like other stakeholders, while she/he holds a key position within the elicitation process. So, it will be better to consider her/him separately.
4. The language used by a stakeholder is a critical factor which directly impacts the requirements elicitation process, since language barriers affect knowledge transfer

¹ see <http://www.engr.ncsu.edu/learningstyles/ilsweb.html>

to and from the analyst. In order to provide the analyst with more information, by considering the set of languages known by each stakeholder participating in a particular iteration of the elicitation process, the role of the *language* parameter can be extended.

Driven by these arguments, we propose to improve the process presented in [3].

3.2 The Proposed Process Model

In the proposed process model (cf. Figure 2), the analyst is able to select an elicitation technique in a cooperative distributed environment based on the following features.

Stakeholder's Classification. We only take into account stakeholders who have strong preferences. It is important to stress here, as shown in section 2, that a stakeholder can be classified in a category only if she/he has strong preferences. Thus, it is needless to consider her/his preferences if she/he has moderate or slightly preferences (because the analyst does not actually know if this stakeholder will feel more comfortable with this elicitation technique).

As Aranda et al. [3], we use Felder-Silverman model [15] which classifies stakeholders as follows: Sensing / Intuitive, Visual / Verbal, Active / Reflective and Sequential / Global. We respectively relate them to the coefficients

$$+C_{11} / + C_{12}, +C_{21} / +C_{22}, +C_{31} / + C_{32}, +C_{41} / +C_{42} \quad (5)$$

$+C_{12}$, $+C_{22}$, $+C_{32}$, and $+C_{42}$ are respectively the coefficients of the opposites subcategories of those related to $+C_{11}$, $+C_{21}$, $+C_{31}$, and $+C_{41}$, we respectively note them: $-C_{11}$, $-C_{21}$, $-C_{31}$ and $-C_{41}$.

For example, if subcategories are: Sensing, Verbal, Active and Global for one particular stakeholder, we represent them with $(+C_{11}, -C_{21}, +C_{31}, -C_{41})$.

Stakeholder's Linguistic Knowledge. Stakeholders' linguistic knowledge contains the set of languages known by each stakeholder. We represent this feature by means of the following structure:

$$\{\text{language}_i (U_i.\text{level}, S.\text{level}, U_r.\text{level}, W.\text{level})\}_{i=1..n} \quad (6)$$

Where:

- $\{\text{language}_i\}_{i=1..n}$ is the set of all the n known languages (e.g. English, Chinese...);
- U , S , U_r and W are performed actions: *Understand_when_listening*, *Speak*, *Understand_when_reading* and *Write* respectively;
- *level* is the knowledge level about the corresponding couple (language_{*i*}, performed action):
- level $\in \{f, m, s\}$ for *Fluently*, *Moderately* and *Slightly* respectively.

For example, a stakeholder *understands when listening fluently*; *speaks moderately*; *understands when reading slightly* and *writes slightly* English language. This is represented by:

$$\text{English } (U.f, S.m, U_r.s, W.s) \quad (7)$$

So, the use of the questionnaire technique (we suppose that the questionnaire is written in English) that requires questions reading and responses writing is not adequate to this stakeholder because of her/his poor ability (knowledge) to read and write English language. The analyst would rather use interview technique in this situation (i.e. the actions *Understand_when_listening* and *Speak* have a higher priority than *Understand_when_reading* and *Write* actions).

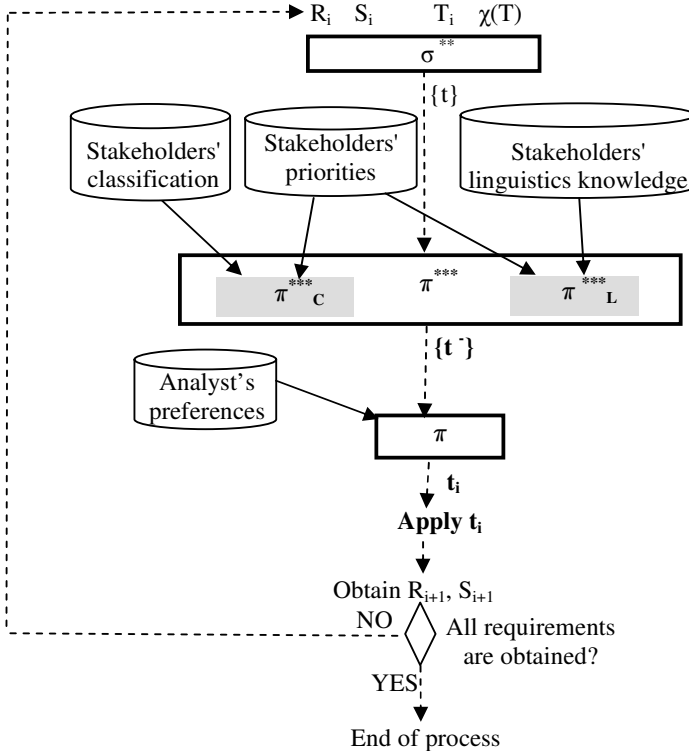


Fig. 2. The different phases of the proposed process

Stakeholder's Priority. Stakeholder's priority value is attributed by the analyst; it depends on stakeholder's role in the elicitation process and/or her/his role in the organization (e.g. complexity of her/his task, whether or not there is another stakeholder who can replace her/him in the elicitation process, i.e. share with her/him the same task, etc). This value is the weight of stakeholder's preferences techniques and linguistic knowledge. The stakeholder's priority is classified from 1 to 5: 1 (very low), 2 (low), 3 (medium), 4 (high), 5 (very high).

In this process, selector function σ^{**} is defined as follows:

$$\sigma^{**}(R_i, S_i, \chi(T), T_i) \rightarrow \{t\} \quad (8)$$

Where:

- $\chi(T)$ is a given set of characteristics of all elicitation techniques T .
- T_i is the degree of overlapping between the different sites: it indicates the possible level of synchronous interaction between the sites. $T_i \in \{\text{no-overlap, little-overlap, half-overlap, much-overlap, full-overlap}\}$. For example, if $T_i = \text{full-overlap}$, then the analyst cannot use a technique which requires a synchronous collaboration between participants, like Brainstorming, but she/he can use the Questionnaire technique.
- $\{t\}$ is a set of elicitation techniques which can be applied in situation S_i when the current state of requirement is R_i according to T_i .

The proposed extension π^{***} of function π^{**} have the following parameters: stakeholders' classification, priorities, and linguistic knowledge.

Function π^{***} is decomposed into two sub-functions π^{***}_C and π^{***}_L (cf. Fig. 2):

- π^{***}_C returns the common classification categories, for stakeholders having strong preferences in the current iteration of the elicitation process, by considering their priorities' values;
- π^{***}_L returns the common linguistic knowledge, for all stakeholders participating in this iteration of the elicitation process, by considering also their priorities' values.

The first sub-function π^{***}_C is defined as follows:

$$\begin{aligned} &\pi^{***}_C ((\pm C_{11}/0, \pm C_{21}/0, \pm C_{31}/0, \pm C_{41}/0), SP_1), \dots, (\pm C_{11}/0, \pm C_{21}/0, \pm C_{31}/0, \\ &\pm C_{41}/0), SP_n)) \\ &= SP_1 * (\pm C_{11}/0, \pm C_{21}/0, \pm C_{31}/0, \pm C_{41}/0) + \dots + SP_n * (\pm C_{11}/0, \pm C_{21}/0, \pm C_{31}/0, \\ &\pm C_{41}/0) \end{aligned} \quad (9)$$

$$\begin{aligned} &= (SP_1 * (\pm C_{11}/0) + \dots + SP_n * (\pm C_{11}/0), SP_1 * (\pm C_{21}/0) + \dots + SP_n * (\pm C_{21}/0), SP_1 \\ &* (\pm C_{31}/0) + \dots + SP_n * (\pm C_{31}/0), SP_1 * (\pm C_{41}/0) + \dots + SP_n * (\pm C_{41}/0)) \end{aligned} \quad (10)$$

$$= (c_1 * (\pm C_{11}/0), c_2 * (\pm C_{21}/0), c_3 * (\pm C_{31}/0), c_4 * (\pm C_{41}/0)) \quad (11)$$

Where:

- SP_k is the k -th stakeholder's priority ;
- $\pm C_{kj}$ is a coefficient related to k -th stakeholder's subcategory classification, 0 indicates that this stakeholder has not strong preferences in this category;
- $c_l = \sum SP_j, j=1..n$ where n is the number of stakeholders having strong preferences in the i^{th} iteration of the elicitation process, for each $l = 1..4$.

Remark 1. The greater c_l in the quadruple (11), the more the elicitation techniques related to the subcategory attached to c_l are taken into account and vice versa.

The second sub-function π^{***}_L is defined as follows:

$$\pi^{***}_L (\{Ls_1\}, SP_1), \dots, (\{Ls_k\}, SP_k), \dots, (\{Ls_n\}, SP_n)) \quad (12)$$

Where:

- LS_k is the set of languages that can be used by the k-th stakeholder followed by the quadruple which defines the level of language's knowledge;
- SP_k is the k-th stakeholder's priority value.

SP_k is multiplied by each quadruple's element that represents the set of languages known by k-th stakeholder. For a language known by more than one stakeholder, we compute the sum of identical elements of the quadruple associated to this language. The result is returned to π^{***} function.

Now, function π^{***} can be defined as follows:

$$\pi^{***}(\{t\}, (11), (12)) \rightarrow \{t^-\} \quad (13)$$

Where:

- (11) is the result of function π^{***}_C ;
- (12) is the result of function π^{***}_L ;
- $\{t\}$ is the set of techniques which results from the function σ^{**} ;
- $\{t^-\} = \{t\} \cap \{\text{techniques related to the subcategory having the coefficient } \pm C_{i1} \text{ attached to } c_1 = \max(c_1, c_2, c_3, c_4) \wedge \text{according to restrictions imposed by (12)}\}$.

Finally the analyst applies function π defined by Hickey and Davis [17], in order to choose one elicitation technique from the set $\{t^-\}$ of techniques returned by function π^{***} . This technique will be applied in the present iteration of the elicitation process.

Function π indicates the priority granted to the analyst at the end of the process. Function π let her/him make the final decision to select one elicitation technique.

Function π is defined as follows:

$$\pi(\{t^-\}, P) \rightarrow t_i \quad (14)$$

Where:

- P is the set of analyst's preferences techniques;
- $\{t^-\}$ is the set of techniques which result from the function π^{***} ;
- t_i is the elicitation technique that is applied in step i of the elicitation process;
 $t_i \in \{t^-\} \cap P$.

Remark 2. The goal of the proposed process is to select only one elicitation technique at the i^{th} iteration of the elicitation process (as Hickey and Davis [17] and Aranda et al. [2], [3] proposed) in a cooperative distributed environment because:

- The analyst often needs to bring together all stakeholders participating in this iteration of the elicitation process and so she/he uses the same elicitation technique (e.g. Brainstorming, Workshop, focus group,...).
- The use of the same elicitation technique facilitates the task of the analyst.

Remark 3. If all stakeholders have not strong preferences or if they have opposite preferences with the same priority (i.e. the result of π^{***}_C is (0, 0, 0, 0)), the analyst

chooses the elicitation technique from the set of techniques $\{t\}$ according only to restrictions of linguistic knowledge π^{***}_L .

4 Two Motivating Examples

4.1 Example 1

In the first example, let us consider the results of the test applied to two stakeholders S_1 and S_2 situated in two distant sites. Stakeholders' characteristics are respectively (Sensing, Verbal, Reflective, 0) for S_1 and (Intuitive, Visual, Active, Global) for S_2 (according to Felder-Silverman classification).

Let us note that:

- S_1 's preference is not strong in the fourth category (the fourth quadruple element = 0);
- S_1 and S_2 have three opposite preferences (Sensing / Intuitive, Verbal / Visual, and Reflective / Active).

Stakeholders' linguistics knowledge is:

- S_1 {English (U_l.f, S.m, Ur.f, W.f), French (U_l.m, S.m, Ur.f, W.m)};
- S_2 {French (U_l.f, S.f, Ur.f, W.f)}.

According to their roles and tasks, their priority is identical, i.e. 1.

We apply the two sub-functions π^{***}_C and π^{***}_L .

$$\begin{aligned} \pi^{***}_C &(((+C_{11}, -C_{21}, -C_{31}, 0), 1), ((-C_{11}, +C_{21}, +C_{31}, -C_{41}), 1)) \\ &= 1*(+C_{11}, -C_{21}, -C_{31}, 0) + 1*(-C_{11}, +C_{21}, +C_{31}, -C_{41}) \\ &= (1*(+C_{11}) + 1*(-C_{11}), 1*(-C_{21}) + 1*(+C_{21}), 1*(-C_{31}) + 1*(+C_{31}), 1*(0) + 1*(-C_{41})) \\ &= (0, 0, 0, -C_{41}) \end{aligned} \quad (15)$$

i.e.

$$c_1 = c_2 = c_3 = 0, c_4 = 1 \quad (16)$$

$$\begin{aligned} \pi^{***}_L &(((\{\text{English (U}_{l\text{.f, S.m, Ur.f, W.f)}, \text{French (U}_{l\text{.m, S.m, Ur.f, W.m)}\}, 1), \\ &(\{\text{French (U}_{l\text{.f, S.f, Ur.f, W.f)}\}, 1)) \\ &= (\{\text{English (1*U}_{l\text{.f, 1*S.m, 1* Ur.f, 1*W.f)}, \text{French (1*U}_{l\text{.m, 1*S.m, 1* Ur.f, 1*W.m)}\}, \\ &\{\text{French (1*U}_{l\text{.f, 1*S.f, 1* Ur.f, 1*W.f)}\}) \\ &= \{\text{English (U}_{l\text{.f, S.m, Ur.f, W.f)}, \text{French (U}_{l\text{.f, m), S.f, m), 2* Ur.f, W.f, m)}\} \end{aligned} \quad (17)$$

Function π^{***} may be defined as follows:

$$\begin{aligned} \pi^{***} &(\{t\}, (15), (17)) = \{t\} \\ &= \{t\} \cap \{\text{elicitation techniques related to the subcategory having the coefficient - } C_{41} \text{ (Global) according to restrictions of linguistic knowledge}\} \end{aligned} \quad (18)$$

The analyst can deduce the two following statements:

- First, she/he must consider the set of techniques related to the subcategory $(-C_{41})$ (Global). The explanation is quite logical: stakeholders' S_1 and S_2 priorities are identical, and their three first subcategories are opposite. Thus the related elicitation techniques attached to these subcategories are ignored.
- In the result of function $\pi_{L,f}^{***}$, the greatest coefficient (i.e. 2 in the present example) is attached to the element $U_{r,f}$ related to French language. So, the use of the questionnaire technique that requires ability in reading questions and writing responses in a specific language (French in this example) can be visualized by the analyst.

4.2 Example 2

Let us take another example, where we attach different priorities to the stakeholders, priority 2 for stakeholder S_1 and 1 for stakeholder S_2 .

The sub-functions π_c^{***} can be defined as:

$$\begin{aligned} \pi_c^{***} &(((+C_{11}, -C_{21}, -C_{31}, 0), 2), ((-C_{11}, +C_{21}, +C_{31}, -C_{41}), 1))) \\ &= 2*(+C_{11}, -C_{21}, -C_{31}, 0) + 1*(-C_{11}, +C_{21}, +C_{31}, -C_{41}) \\ &= (2*(+C_{11}) + 1*(-C_{11}), 2*(-C_{21}) + 1*(+C_{21}), 2*(-C_{31}) + 1*(+C_{31}), 2*(0) + 1*(-C_{41})) \quad (19) \\ &= (+C_{11}, -C_{21}, -C_{31}, -C_{41}) \end{aligned}$$

i.e.

$$c_1 = c_2 = c_3 = c_4 = 1 \quad (20)$$

So, we can take into account elicitation techniques related to any one of the subcategories $+C_{11}$, $-C_{21}$, $-C_{31}$ and $-C_{41}$ (i.e. Sensing, Verbal, Reflective and Global). Because $+C_{11}$, $-C_{21}$, $-C_{31}$ are related to the stakeholder with the highest priority, then the use of elicitation techniques related to these subcategories is justified; $-C_{41}$ is related to the stakeholder with the lowest priority, but the preferences attached to this subcategory have not negative influence on the other stakeholder, because she/he is not Sequential, so the use of elicitation techniques attached to this subcategory is understandable.

5 Related Works

Much of the research efforts in requirements engineering have been focusing on the requirements themselves: how to elicit them, analyze them, how to resolve conflicts and manage them and so on. But in the literature, few works have been focusing on the selection of an elicitation technique, especially in a distributed environment.

Several studies identified a broad range of challenges related to the requirements engineering process in a distributed environment. Much of these studies identified problems related to cultural diversity, process and tools, time difference between sites, remote communication and knowledge management, and their negative impact on requirements elicitation, negotiation and specification in a distributed environment [9], [10], [11], [11], [16], [20], but they did not deal with these challenges.

For effective requirements elicitation in a cooperative distributed environment, it is worth exploring additional issues which impact the product quality and, in particular, to significantly improve analyst's ability to select the appropriate elicitation technique in a given situation of the requirement elicitation process.

As presented in the section 2.3, Hickey and Davis [17] introduced a new model of requirements elicitation, which defines the underlying basis of an implementation of the elicitation technique selector function. On the basis of these results Aranda et al. [2] [3], proposed a process which can be used to select an elicitation technique in a distributed environment. They introduced new concepts, from cognitive informatics in order to consider stakeholder's characteristics, the time difference between different sites and the level of knowledge of a common language. However this process (1) does not take into account the case where stakeholders' preferences are opposite, and (2) gives a general representation of stakeholders' linguistic knowledge.

As we early explained, the process we propose aims to facilitate the analyst's task in selecting an elicitation technique in a cooperative distributed environment; by considering the diversity of stakeholders' priorities, preferences, linguistic knowledge and time difference between distant sites.

To achieve this goal:

- we attach opposite coefficients to each opposite subcategories;
- we attach a number value to each stakeholder's priority;
- we introduce a structured representation of the linguistic knowledge level.

6 Concluding Remarks and Further Work

Computer-supported cooperative work and requirements elicitations are two currently up to date fields of interest. The former comes out of the necessity for companies to cooperate within an accentuated competitive context. As for the latter, it is based on a set of techniques provided to an analyst in order to build the first stage of an information system design. We presented in this paper a process to improve requirements elicitation, based on a model which helps the analyst to choose easily and efficiently the elicitation technique adapted to a given situation of the requirements elicitation process.

The process we propose roughly seems to be a self-inflicted wound. Why to adopt it, if it gives rise to an apparently gratuitous additional difficulty? The answer is based on the following empirical evidence:

- A stakeholder feels comfortable with some elicitation techniques, and she/he feels uncomfortable with others. These preferences are issued from stakeholder's characteristics and they allow elaborating a classification of stakeholders.
- In a cooperative environment, either distributed or not, stakeholders' linguistic knowledge may be different, which affect the global collaboration.
- Stakeholders' roles and tasks are different in their organization within the requirements process. Then their priorities may be also different, and they must be taken into account by the analyst when selecting a requirements elicitation technique.

- The time difference between distant sites can allow or not an overlap for synchronous collaboration, which implies the possibility to use or not some elicitation techniques.

In short, the proposed process helps to improve the quality of the requirements elicitation process, by taking into account that the previous features have a significant impact on the choice of a requirements elicitation technique in a distributed cooperative environment.

As an extension to the current work, we plan to perform two actions:

- An implementation of the used functions. This requires the consideration of several factors like project and solution situations, requirements state and stakeholders' characteristics.
- Apply the proposed process to more case studies and real-life software projects.

References

1. Aranda, G., Cechich, A., Vizcaíno, A., Castro-Schez, J.J.: Using fuzzy sets to analyse personal preferences on groupware tools. In: X Congreso Argentino de Ciencias de la Computación, CACIC 2004, San Justo, Argentina, pp. 549–560 (2004)
2. Aranda, G., Vizcaíno, A., Cechich, A., Piattini, M.: A Cognitive-Based Approach to Improve Distributed Requirement Elicitation Processes. In: 4th IEEE International Conference on Cognitive Informatics (ICCI 2005), Irvine, USA (2005)
3. Aranda, G., Vizcaíno, A., Cechich, A., Piattini, M.: A Cognitive Perspective for Choosing Groupware Tools and Elicitation Techniques in Virtual Teams. In: Gervasi, O., Gavrilova, M.L., Kumar, V., Laganá, A., Lee, H.P., Mun, Y., Taniar, D., Tan, C.J.K. (eds.) ICCSA 2005. LNCS, vol. 3480, pp. 1064–1074. Springer, Heidelberg (2005)
4. Aranda, G., Vizcaíno, A., Cechich, A., Piattini, M.: Towards a Cognitive-Based Approach to Distributed Requirement Elicitation Processes. In: Workshop em Engenharia de Requisitos, Porto, Portugal (2005)
5. Bendjenna, H., Zarour, N., Charrel, P.J.: Elicitation Technique Selection in a Cooperative Distributed Environment: How analysts make the best decision. In: International Conference on Rapid Integration of Software Engineering techniques, RISE, Luxembourg, pp. 32–47 (2007)
6. CHAOS report: Standish group (1995), http://www.standishgroup.com/sample_research/chaos_1994_1.php
7. Chiew, V., Wang, Y.: From Cognitive Psychology to Cognitive Informatics. In: Second IEEE International Conference on Cognitive Informatics, ICCI 2003, London, UK, pp. 114–120 (2003)
8. Damian, D., Lanubile, F., Hargreaves, E., Chisan, J.: Workshop Introduction. In: 3rd International Workshop On Global Software Development, Co-located with ICSE 2004, Edinburgh, Scotland (2004)
9. Damian, D., Zowghi, D.: The impact of stakeholders' geographical distribution on managing requirements in a multi-site organization. In: IEEE Joint International Conference on Requirements Engineering, RE 2002, Essen, Germany, pp. 319–328 (2002)
10. Damian, D.E., Zowghi, D.: An insight into the interplay between culture, conflict and distance in globally distributed requirements negotiations. In: 36th IEEE Hawaii International Conference on System Sciences (HICSS 2003), 0-7695-1874-5/03 (2002)

11. Damian, D.E., Zowghi, D.: Requirements Engineering challenges in multi-site software development organizations. *Requirements Engineering Journal* 8, 149–160 (2003)
12. Damian, D., Hadwin, A., Al-Ani, B.: Instructional design and assessment strategies for teaching global software development: a framework. In: 28th International Conference on Software Engineering, Shanghai, China, pp. 685–690 (2006)
13. Davis, A.: *Software Requirements: Objects, Functions and States*. Prentice Hall, Upper Saddle River, New Jersey (1993)
14. Davis, A., Hickey, A.M.: Requirements Researchers: Do We Practice What We Preach. *Requirements Engineering Journal* 7(2), 107–111 (2002)
15. Felder, R., Silverman, L.: Learning and Teaching Styles in Engineering Education. *Engineering Education* 78(7), 674–681 (1988)
16. Herbsleb, J., Paulish, D., Bass, M.: Global Software Development at Siemens: Experience from Nine Projects. In: 27th International Conference on Software Engineering, St. Louis, Missouri, USA, pp. 524–533 (2005)
17. Hickey, A.M., Davis, A.: Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes. In: 36th Annual Hawaii International Conference on Systems Sciences (HICSS), pp. 96–105 (2003)
18. Hickey, A.M., Davis, A.: Elicitation Technique Selection: How Do Experts Do it? *Requirements Engineering 2003*, 11th IEEE International Volume, 169–178 (2003)
19. Leffingwell, D., Widrig, D.D.: *Managing Software Requirements: A Unified Approach*. Addison Wesley Publishing Co., Reading (2000)
20. Lloyd, W., Rosson, M.B., Arthur, J.: Effectiveness of elicitation techniques in distributed requirement engineering. In: 10th anniversary IEEE Joint international conference on requirement engineering RE 2002, Essen, Germany, pp. 311–318 (2002)
21. Macaulay, L.: *Requirements Engineering*. Springer, Heidelberg (1996)
22. Reubenstein, H., Waters, R.: The Requirements Apprentice: Automated Assistance for Requirements Acquisition. *IEEE Transactions on Software Engineering* 17(3), 226–240 (1991)
23. SWEBOK. Guide to the Software Engineering Body of Knowledge. Software Engineering Coordinating Committee. IEEE Computer Society, Los Alamitos (2004)
24. Thayer, R., Merlin, D.: *Software Requirements Engineering*, 2nd edn. IEEE Computer Society, Los Alamitos (1997)
25. Van Buren, J., Cook, D.: Experiences in the Adoption of Requirements Engineering Technologies, CROSSTALK. *The Journal of Defense Software Engineering* 11(12), 3–10 (1998)
26. Wiegers, K.E.: *Software Requirements*, 2nd edn. Microsoft Press (2003) (1st edn. - 1999)
27. Williams, W.: What Do You Mean You Can't Tell Me If My Project Is in Trouble? In: First European Conference on Software Metrics (FESMA 1998), Antwerp, Belgium (1998)

Negotiation Constellations – Method Selection Framework for Requirements Negotiation

Samuel Fricker^{1,2} and Paul Grünbacher³

¹ University of Zurich, Department of Informatics
Binzmuehlestrasse 14, 8057 Zurich, Switzerland
fricker@ifi.uzh.ch

² ABB Switzerland Ltd., Power Systems
Bruggerstrasse 72, 5400 Baden, Switzerland
samuel.fricker@ch.abb.com

³ Johannes Kepler Universität (JKU),
Institute for Systems Engineering and Automation
4040 Linz, Austria
paul.gruenbacher@jku.at

Abstract. Customers, product managers, project leaders, architects, engineers, and other stakeholders are negotiating requirements throughout the software lifecycle. Even though fundamental for understanding requirements engineering, negotiation has not been as thoroughly studied as other facets of this engineering discipline. This paper casts requirements engineering into the landscape of negotiation by describing a framework for selecting tactics and methods for various negotiation constellations that can be encountered in a software organization. The framework opens perspectives that are essential for understanding the behavior of people involved in development projects, for understanding how development teams and stakeholders create mutually satisfactory solutions, and for giving tactical advice to practitioners.

1 Introduction

Software development is embedded in a complex network of stakeholders that include roles like customers, development managers, product managers, team leaders, architects, developers, testers, and maintainers [10]. The interplay between these stakeholders is a fundamental success factor, as every role brings essential knowledge, capabilities, and skills that are essential to design great new products.

However, designing appropriate requirements engineering processes for such complex stakeholder networks is still a major challenge [7]. For instance, there are multiple organizational interfaces at which requirements are engineered: it can be observed that stakeholders pursue their own objectives by trying to delegate the fulfillment of some goals while satisfying those of others [33]. This happens not only during early-phase requirements engineering, but also in design and change management activities throughout the whole development process [5,11].

The lack of approaches for tailoring requirements engineering to the structure of stakeholder networks and to the specific negotiations situations between these stakeholders

leads to misunderstandings and conflicts. As a consequence, development effort is wasted on insignificant features, rather than being invested on features that are most essential for stakeholder satisfaction.

In response to these challenges posed by complex stakeholder networks, this paper presents a framework for helping stakeholders to understand their *negotiation constellations* and for *selecting appropriate negotiation tactics and methods*. The proper selection of negotiation tactics and methods enables effective communication and acknowledgment of requirements, helps exploiting opportunities for stakeholder satisfaction by creating win-win situations, and establishes trust relationships that are important for development efficacy and high-impact development results.

Beyond its usefulness for practitioners, we hope that the framework will aid requirements engineering researchers to structure and understand the landscape of negotiation in requirements engineering. The framework references knowledge from the broad field of negotiation and identifies a number of research opportunities for understanding on how to handle requirements adequately in specific stakeholder constellations.

This paper presents the negotiation constellations framework and its implications on requirements engineering practice and research. The paper is structured as follows: Section 2 outlines background and related work. Section 3 presents the negotiation constellation framework. Section 4 illustrates the use of the framework. Section 5 discusses the presented work. Section 6 summarizes and concludes the paper.

2 Background and Related Work

This work has been motivated by challenges identified at ABB. It relates to a case where product managers coordinate distributed development teams with requirements that are derived from agreements with diverse stakeholders. Conflicts arise almost inevitably in such cases as project stakeholders pursue mismatching goals and try to influence each other [12,19]. For example, in a software product organization goals need to be considered from the market, partners, customers, users, company management, sales & marketing, research & innovation, consultants, development, and support [1,32]. Successful requirements engineering demands agreement on the requirements [15].

Key approaches that can be applied to reach such an agreement include analysis of viewpoints [14], stakeholder and goal modeling [15,33], and negotiation [6,12,16,31]. The negotiation process starts when the stakeholders communicate their goals. It ends when all have agreed to a specified contract [26].

There are two fundamental ways to manage this negotiation process with regard to how agreements are established in the stakeholder network. First, the process can be managed by a requirements engineer who elicits the positions and perspectives of stakeholders, documents them in a comprehensive goal model, facilitates the resolution of conflicts, and communicates the obtained global stakeholder agreement.

Second, the negotiation process can emerge out of the activities of stakeholders that perform the organizational roles they are assigned to. Instead of one large negotiation that involves all stakeholders, negotiation is carried out as a number of

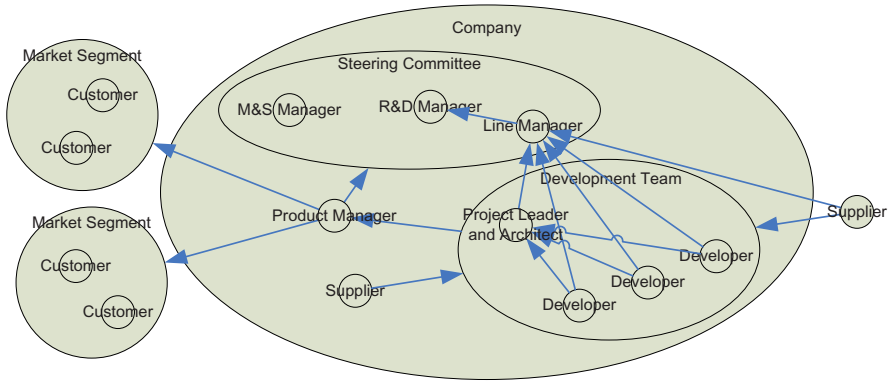


Fig. 1. Exemplary contract model of a software organization. Ellipses represent hierarchically nested groups of people or individuals. Arrows represent contracts that are agreed upon.

small-scale activities that are performed rather independently. This leads to a number of agreements between different stakeholders [7]. An example of such distributed negotiations is illustrated by the contract model shown in Fig. 1.

Independent of the process flavor, questions about the tactical approach and methodology appear in these different negotiation constellations. Requirements engineers need to understand how to perform win-win negotiations, how to reach value-creating results, and how to deal with group dynamics. Stakeholders need to understand their role in the negotiation process and what they can and should do to achieve their objectives by influencing other stakeholders. Hence, the following issues need to be addressed:

- Correctly conceptualizing the negotiation constellation,
- Understanding the advantages and limitations of the constellation,
- Knowing the negotiation tactics and methods appropriate for the constellation,
- Identifying those stakeholders that need to be involved in negotiation, and
- Selecting and pursue the most appropriate negotiation approach.

The knowledge in the *negotiation constellations framework* assists stakeholders with these questions and provides negotiation advice. It also is used to improve requirements engineering processes by capturing, organizing and making available good negotiation practices and experiences.

The negotiation constellations framework is similar to reference models like CMMI for software process improvement [13], and the good practice guide for requirements engineering improvement [29]. It focuses, however, on requirements negotiation and adds criteria for selecting tactics and methods that are based on the situations in which they are applied. In contrast to other reference models, the negotiation constellations framework also supports capturing and structuring experience to support learning software organizations [28].

3 Negotiation Constellations

Understanding negotiation constellations is essential for efficiently finding good agreements among stakeholders. This section elaborates how the negotiation constellation framework advises practitioners and supports requirements engineering process improvement by describing its structure and use.

Negotiation is an interpersonal decision-making process to find a mutually acceptable agreement to a conflict [16,31]. Agreements can contain the planned realization of needs and objectives, the use of capabilities, the guarantee of financial or other backing, or the provision of knowledge [8].

A *negotiation constellation* is characterized by a number of facets that influence the selection of negotiation methods. Key facets include the characteristics of the negotiating parties, the relationships between these parties, and the negotiation object [31]. Other facets include the geographical distance between the parties [6] and their expected conflict behavior [30].

The *negotiation constellations framework* describes a taxonomy of negotiation constellations and provides specific advice for negotiation tactic, methodology, and experience for a given negotiation constellation. The framework was shaped to be relevant, simple, specific, and orthogonal. It contains knowledge that is useful for advising practitioners in a software development context. The number of taxonomic units is intentionally kept small. The decision criteria are simple and can be applied intuitively. The advice is given at a coarse level of granularity that still allows differentiating negotiation approaches. The number of fields in which the same negotiation tactics and techniques are found is minimized, however without compromising specificity.

The negotiation constellations framework has been defined with the following research process in collaboration with practitioners. Situations have been identified that require applying different negotiation tactics and techniques. These situations were then exemplified with stereotypical descriptions of software development organizations and relationships between various organizational roles. Finally, negotiation, requirements engineering and software engineering literature was studied to identify tactics and methods that adequately address the negotiation situations.

Subsection 3.1 describes commonalities of negotiation situations in a software engineering context. Subsection 3.2 describes the taxonomy of negotiation constellations. Subsections 3.3 and 3.4 describe tactical and methodological advice.

3.1 Common Negotiation Characteristics in Software Organizations

Negotiation has been studied in many different contexts, including product sales, employment contracts, personal affairs, politics, and peace keeping. Negotiation occurs 1) to agree on how to share or divide limited resources such as money, time and staff; 2) to create something new that neither party could do on its own; or 3) to resolve a conflict between parties. By choosing options other than negotiation, people may fail to achieve their goals, get what they need, or manage conflicts as smoothly as they might like to [16].

Negotiation in a software organization is special because a number of factors in the negotiation context are predetermined. This significantly reduces the variability of

general negotiation situations and allows simplifying the negotiation constellation framework. The factors that are specific to software organizations concern the negotiation object, conflict management, and opportunities for renegotiations.

Bargaining over a *single issue* like a price is rare. Instead, people seek win-win results that occur when a mutually acceptable solution is sought. Win-win negotiation involves a *number of issues that are negotiated together*. For instance, a customer may want to reduce the price of a software solution or service, but this is typically negotiated together with other contractual elements like the scope of the solution or service. In other circumstances, people negotiate a set of concerns and objectives such as needs, requirements, and design decisions.

A number of conflict resolution styles are differentiated in negotiation, depending on the negotiators interest in his own outcome and in the other negotiator's outcome [27]. This dual-concerns model is illustrated in Fig. 2.

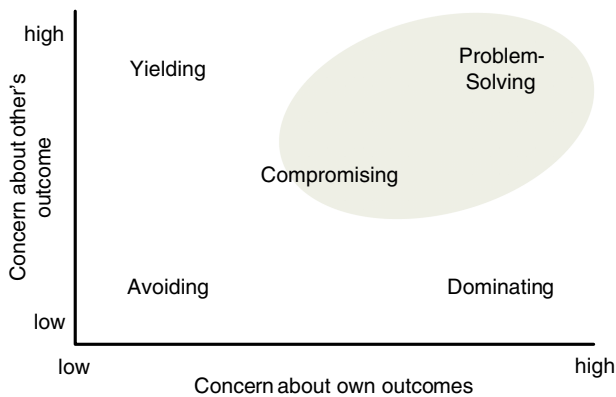


Fig. 2. Dual-concerns model of negotiation behavior [27]. The grey area highlights the conflict resolution style in a software organization centered on problem solving or compromising.

The conflict resolution style that should preferably be adopted in a software organization is *problem solving*, or *compromising* when consensus cannot be reached [23]. The issues that are negotiated in a software organization are complex: a synthesis of ideas is needed to come up with mutually satisfactory solutions, and time is available for such problem solving. Resources, skills and knowledge are possessed by different parties. Hence, commitment is needed from these other parties for successful implementation, with one party alone not being able to solve the negotiated problems. *Yielding* to another party should not be done because the issues negotiated are important, in the responsibility of the negotiators, and ultimately connected to their career. For the same reason, *avoiding* the other party is inappropriate. Finally, the other party should not be *dominated* because the negotiated issues are too complex and the negotiation partners have high degree of competence in their areas.

In software organizations, a number of *opportunities for renegotiation* are institutionalized. For example, change management processes are established to

account for imperfect design and technology evaluation and planning. Hence, agreements are not carved in stone and may be changed. Still, the negotiators should be concerned about their reputation, because excessive and late use of renegotiation may severely weaken their position as an accepted negotiation partner.

The generic negotiation tactic in a software organization is integrative negotiation: be prepared, create value, and claim your share of the created value [31].

During *preparation* a negotiator¹ assesses his aspirations, his best alternative to a negotiated agreement (BATNA), and his reservation point at which he would stop the negotiations. He tries to elicit the same information from the negotiations partners by possibly revealing his aspirations, but without disclosing his BATNA and reservation point. In addition, he takes the situational factors into consideration that are described by the negotiation constellations framework.

Value creation can be achieved with creative conflict resolution. Good ideas can be identified when the negotiators trust each other, share information, and adjust the negotiation issues. Value can be created by capitalizing on differences in the valuation or preferences for goals, the forecast of the future, risk attitudes, time preferences, and capabilities. For example, a product marketing manager wanting to realize a number of product features may be faced with different design ideas by a development team of how such features can be implemented. The negotiation will cover a stage where the design ideas are created and evaluated by these parties.

In the late stage of a negotiation, the negotiators increasingly *claim value*. A negotiator claims value by a steadily improving its BATNA, anchoring the negotiation in the area of its aspirations, and planning for a sequence of concessions. To support value claiming, he can appeal to a number of facets to fairness, including equality, where all should get equal shares, equity, where the share is proportional to the party's contribution, and need, where share is proportional to the party's need.

Fig. 3 presents a model that explains the interrelationships of creating and claiming value in multi-issue negotiations [24].

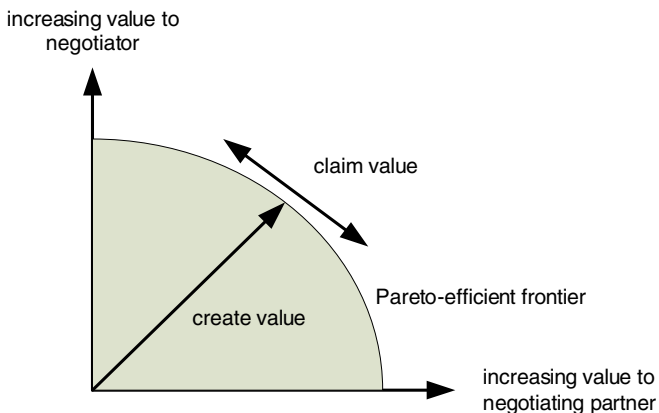


Fig. 3. Conceptualization of creating and claiming value [16]. Maximal value is created when a point on the Pareto-efficient frontier is reached.

¹ For legibility reasons, we use the term 'he', but mean both sexes.

3.2 Characterization of the Negotiating Parties

To select appropriate negotiation tactics and methods, a negotiator needs to know his and his negotiation partner's constitutions. Fig. 4 shows the taxonomy of such constitutions, which is fundamental to the negotiation constellation framework.

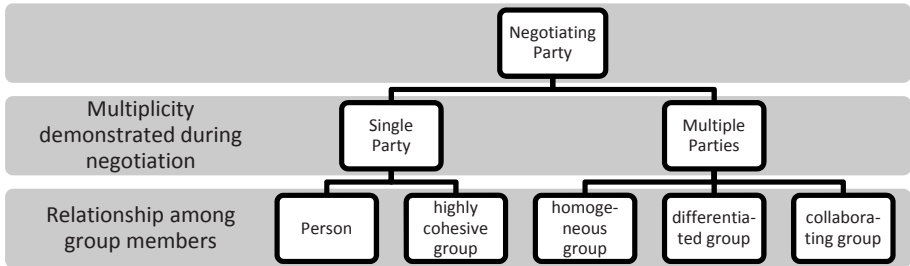


Fig. 4. Constitutions of negotiating parties shown with 'is-a-kind-of'-refinements

A *single party* is a person or a highly cohesive group of people. A single party has one set of aspirations, one BATNA, one reservation point, and one voice at the negotiation table. No internal fragmentation exists: there is neither intrapersonal conflict of the person nor interpersonal conflict in the cohesive group of people.

Typical roles of individual people in a software organization are line manager, product marketing manager, project manager, or architect. Examples of groups of people that appear as a single party at a negotiation table are a company in the role of a customer or supplier, management of a company when negotiating with employees, and a development team when negotiating with stakeholders.

The differentiation between a person or a highly cohesive group of people is not further used in the negotiation constellations framework. Both should use the same negotiation tactics and processes during a negotiation. It is likely, however, that in the course of software development, a group may recognize that it is not as cohesive as perceived initially. This can lead to a different negotiation situation and may require switching the mode of negotiation.

Multiple parties are a group of people that appears at one side of the negotiation table. In contrast to the single party, the constitution of the group is important. The group can consist of single parties or again other groups. The multi-party group is characterized by at least one of the following properties: the group members pursue different objectives, have different BATNA and reservation points, and have individual voices at the negotiation table. Since group members are differentiated, agreements made at the primary negotiation table should be ratified.

Typical examples of multiple parties are companies that make up a market, software users, management, a steering committee when negotiating with a project manager, an architecture team when negotiating with a product marketing manager, and a project team when negotiating with its project manager.

For the purpose of negotiation tactic and method selection, homogeneous groups, differentiated groups, and collaborating groups are distinguished. *Homogeneous*

groups consist of members that have the same aspirations, BATNA and reservation point, but have individual voices. All members are willing to comply with negotiation results that are equal for everyone. A typical example is users within a user group.

A member of a *differentiated group* has, in addition to an individual voice, the desire to be different from the other group members. This leads to different aspirations, BATNA, and reservations points. Members of such a group are often competing with each other. A typical example is technology suppliers.

Members of a *collaborating group* also have individual voices, aspirations, BATNA, and reservation points. In contrast to the differentiated group, they seek an agreement that is satisfactory for every member. Rather than being in competition, the members of a collaborating group have different perspectives on the negotiation topic and have complementing aspirations, knowledge, networks, and capabilities.

3.3 Micro-level: Negotiation Tactics

The objective of the negotiation constellations framework is to help people in a software context to negotiate better. At a micro-level, the framework offers partisan tactical advice to a negotiator at the possible expense of his negotiation partner. Still, this advice is fair, because it is open for everyone. At the macro-level the framework offers methodological advice that helps all involved parties.

The *tactical negotiation constellation framework* differentiates between the negotiator who benefits from the advice and his negotiating partners. Both are involved in a negotiation that ultimately results in decisions about requirements, project plans, architectural design, and the like. The framework allows the negotiator to understand his negotiation constellation in terms of who he is and who the other is, and suggests tactical actions that strengthen his negotiation position.

The tactical negotiation constellations framework is shown in Fig. 5. The presented tactical advice is based on standard negotiation textbooks [31].

		Partner(s)			
		Single Party	Multiple Parties		
			Homogeneous	Differentiated	Collaborating
Yourself	Single Party	Constituent	Constituent	Select	Coalition
	Multiple Parties	Principal Agent, Constituent	Principal Agent, Constituent	Principal Agent, Select	Principal Agent, Coalition
		Differentiated	act as single party		
	Collaborating	Principal Agent, Team Negotiation, Constituent	Principal Agent, Constituent	Principal Agent, Select	Coalition, Intergroup Negotiation

Fig. 5. Tactical advice for different negotiation constellations. Section 4 exemplifies.

The advice can be read out from the negotiation constellations framework by consulting the cell that corresponds to the negotiator’s perception of himself and of his partner. For example, if the negotiator is a single party, with multiple homogeneous partners, he can increase the value of what he gets or speed up the negotiations by influencing the partners through constituents.

While the provided advice is specific to the constitution of the both negotiators, the framework shows that some decisions depend only on the constitution of the negotiator, and other decisions on the constitution of the negotiation partner. For example, homogeneous parties can be influenced with a constituent, independently of the structure of the primary negotiator. Acting as a single party helps the negotiator who is in competition with peers, independent of the negotiating partner.

Table 1 explains the tactics suggested by the tactical negotiation constellations framework. A discussion of the advantages and risks of using the negotiation tactics can be found in standard textbooks [31].

Table 1. Explanation of negotiation tactics

Tactic	Explanation
Constituent	The use of peripheral players that have an indirect stake in the outcome to exert pressure on the other side.
Select	Stick to the party with the most promising outcome.
Coalition	Exert influence on outcomes by collaborating with a minimal but sufficient number of partners.
Principal Agent	Use an experienced agent to prepare or to run the negotiations on behalf of yourself.
Team Negotiation	Prepare and run the negotiations as a team to increase creativity and control of the negotiation.
Intergroup Negotiation	Control the conflicts that naturally appear in the confrontation of two or more groups.

3.4 Macro Level: Negotiation Methods

On a macro level, the negotiation constellations framework suggests methods and processes that maximize the value of the outcome and the satisfaction of the negotiators. The *methodological negotiation constellation framework* differentiates between generalized customer and supplier roles that engage in negotiations, without losing generality compared with the tactical framework. Fig. 6 shows the framework.

		Supplier			
		Single Party	Multiple Parties		
			Homogeneous	Differentiated	Collaborating
Customer	Single Party	<i>Handshaking</i>	<i>Plug-In Architectures</i>	<i>COTS Selection</i>	<i>Team Problem Solving</i>
	Multiple Parties	<i>MD-RE</i>	Standardization	Survival of the Fittest	New Product Development
		<i>Domain RE</i>	Socialist Markets	Competitive Markets	<i>Product Line Engineering</i>
		<i>VORD</i>	Governance	Voting, Consensus...	<i>EasyWinWin</i>

Fig. 6. Methodological advice for different negotiation constellations. Italic entries refer to approaches from requirements or software engineering. The other entries describe metaphors for the negotiation constellations.

In addition to the self-assessment and the assessment of the negotiation partners, the negotiator analyzes the relationship to understand who is in a customer, supplier, or peer role. If he is in a customer role he places himself on a row, otherwise on a column. If some negotiators are peers, he and they together form a multiparty.

The method framework reflects in its basic form the state of knowledge. This implies that one, several, or no published methods can be identified for the various negotiation constellations. This advice should evolve by new research results and by the experiences made by those using it.

For example, for the one customer – one supplier constellation, one well-fitting method could be identified. The two parties will reach the best results if handshaking [24] is adopted for negotiation.

For the one customer – differentiated suppliers constellation, several methods could be identified. As long as the principles underlying these methods are not elaborated from the specific perspective of the negotiation situation, the negotiators have to select the best-fitting method. Such selection needs to be based on a refined understanding of the issues that are negotiated and the capabilities of the candidate methods. For example, to procure COTS software from candidate suppliers, a customer will employ one of the many supplier and COTS selection methods [17].

For the differentiated customers – differentiated suppliers constellation, fitting methods are hard to find. Here the framework only indicates a metaphor for approaching the situation. For example, to describe the behavior of customers in a segmented market confronted with a number of software suppliers, the laws of competitive markets apply [21].

Table 2 references methods for those cells of the methodological negotiation constellations framework, for which methods could be identified. These methods represent the initial recommendations that are evaluated for the given negotiation constellation and adjusted as experience and improved state of knowledge suggest.

Table 2. Methods fitting the various negotiation constellations

Method	Short Description
Handshaking	The use of implementation proposals to control understanding of communicated requirements [24].
Plug-in Architecture	Software design for extensibility by defining consistent ways and means of third-party software integration [18].
COTS Selection	The use of criteria for selecting commercial off-the-shelf products for system development [3].
MD-RE	Market-driven requirements engineering addresses the management of requirements for a number of customers [25].
Domain-RE	Identify and analyze common and variable requirements [20].
Product Line Engineering	Develop software for heterogeneous needs [20].
VORD	The capture, analysis and resolution of different needs and ideas with viewpoints [14].
EasyWinWin	Multi-party requirements negotiation approach [2].
Team Problem Solving	Defining solutions to problems in a team [17].
Standardization	Establish a consistent technical specification for a number of players [22].
New Product Development	Coordination of roles for new product development [4].

4 Framework Use for Tactical and Methodological Advice

This section illustrates the use of the negotiation constellations framework based on the company described in Fig. 1. The illustration follows a narrative that is inspired by experienced practice. Careful empirical evaluation, however, is ongoing work.

The narrative and the contract models in Fig. 7 describe how various roles in the software organization perceive their negotiation context and use the negotiation constellations framework for advice on how to proceed tactically and methodologically. As such perception is highly personal, the decisions by the players represent just one of many possible courses of actions.

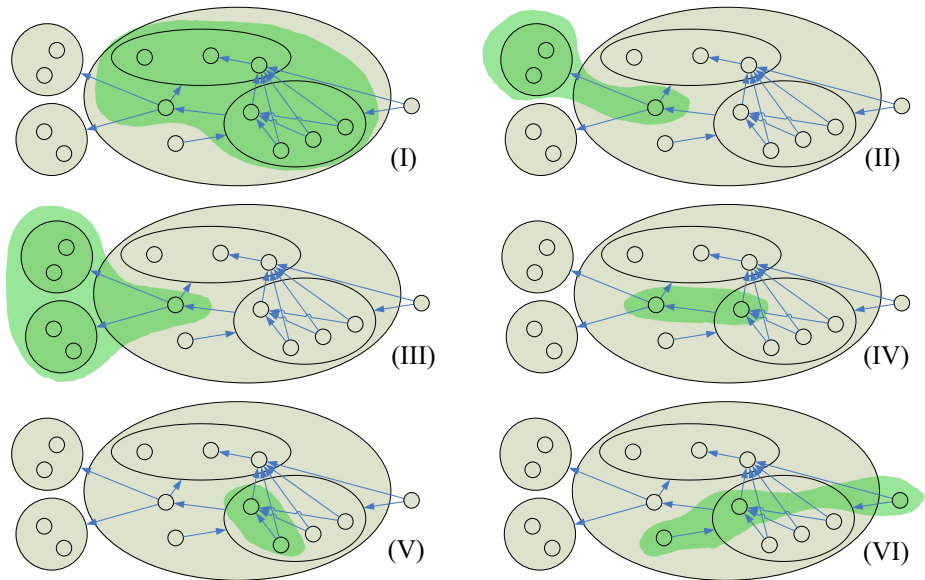


Fig. 7. Negotiation constellations, highlighted as shaded areas, in the organization described in Fig.1. The negotiation constellation framework provides tactical and methodological advice for such constellations.

(I) The *project leader and architect*, a member of the *development team*, is responsible for establishing architectural decisions that satisfy the needs represented by the stakeholders *product manager* and *steering committee* and for committing *developers* to implement the software according to these decisions. In this situation, he is confronted with a number of collaborating customers, the stakeholders, and a number of suppliers, himself and the *developers*. The negotiation constellations framework suggests using EasyWinWin as a methodology, building coalitions, and dealing with intergroup negotiation issues.

(II) The *product manager* is responsible to understand the company's *markets* and to identify requirements that best address the *customers'* needs. In this situation he may look at the market as a single *market segment* with homogeneous *customer*

needs. Here he is well-advised with market-driven requirements engineering as a methodology.

(III) Alternatively, the *product manager* may identify multiple *market segments* with differentiated groups of *customers*. In this situation he is better advised to follow a domain requirements engineering approach for better understanding the variability of the needs of the different segments.

(IV) At some moment, the *product manager* has produced a software requirements specification that he hands over to the *project leader and architect* of the *development team*. The *development team* sees itself as a number of collaborating people and decides to use the *project leader and architect* as a principal agent, as suggested by the tactical negotiation constellations framework. The requirements hand-over situation, thus, is reduced to a negotiation between two individuals that is best addressed by handshaking with implementation proposals.

(V) To further progress in the implementation of the software, the *project leader and architect* conveys architectural decisions and distributes tasks to individual *developers*. Here the advice is again to use handshaking.

(VI) Finally, the *project leader and architect* sees opportunities to speed up development work with components that can be procured from an in-house or from an external *supplier*. Here he is confronted with a selection task where he adopts COTS-selection as a method.

5 Discussion

5.1 Practical Considerations

Section 4 has shown how the negotiation constellations framework can be used to provide tactical and methodological advice in practical situations. It helps a person or organization to conceptualize a negotiation constellation, to understand the advantages and limitations of the constellation, to know which tactics and methods are appropriate, and to identify the stakeholders that should be involved. Hence, the framework helps to exploit the strengths of the negotiation constellation and to understand its limitations.

As above illustration has shown, negotiation in a software context is not a one-shot activity. Rather, a sequence of overlapping negotiations is performed in practice. These negotiations are overlapping in time and in the people that are involved. Skilled negotiators do not act passively, but proactively try to shape the negotiation constellations in an attempt to strengthen their negotiation position for increasing the chances to achieve their objectives. The negotiation constellations framework may evolve into a valuable tool to support such reflections and is a basis for shaping and describing negotiation strategies.

As people and organizations enact the negotiation tactics and methods, they gain experiences, which can be reused [28]. The negotiation constellations framework provides a structure and means for such reuse. When advice has worked well, it is supplemented with experience data. When tactics or methods have been discovered that fit the negotiation constellation better in the specific negotiation constellation, previous advice is replaced by improved advice.

5.2 Implications on Research and Education

In addition to practical benefit, the negotiation constellations framework opens a number of perspectives for research and education. The framework provides a structured approach to transfer knowledge from the field of negotiation into requirements and software engineering. The table cells refer to specialized negotiation literature through the named tactics.

The framework organizes knowledge based on simple criteria that are relevant for practice. It is thus a basis to study the applicability of tactics and methods by comparing the organizational contexts which they apply to.

In the same line, the negotiation constellations framework helps to better understand limitations of current knowledge in requirements engineering. While all cells are relevant, for some negotiation constellations it is hard to find focused requirements or software engineering methods.

Finally, negotiation has the potential to act as a model of how requirements are communicated and transformed into design decisions. A better understanding of negotiation in the software context will lead to a better understanding of the co-evolution of requirements and design.

5.3 Limitations

The negotiation constellations framework has been designed for simplicity. This may be in conflict with the complexity of the real-world situations, where it is intended to be used. Experienced skillful negotiators act in a much more multi-faceted manner than the negotiation constellation framework suggests by adjusting to factors like geographical distance and negotiation style. Also a negotiator is typically embedded into a complex network of partners, which is not represented by the simple customer-supplier relationship of the framework. Still the negotiation constellation framework is a useful starting point for companies that wish to address requirements negotiation in a systematic manner.

The tactical and methodological advice that is suggested by the negotiation constellations framework is incomplete and requires adaptation to an organization. If consensus on the superiority of a given negotiation approach is not possible, the negotiation constellations framework needs to be tailored to parts of the company, or even to a single role. The framework would still be useful for providing advice and capturing experience, but a number of instances will need to be managed.

The fields of negotiation, requirements engineering and software engineering are evolving. This is an opportunity for the framework to mature, as more specific tactics and methods are discovered. With the evolving fields, the knowledge that is stored in the framework can be completed and improved.

The research on the negotiation constellation framework is still in progress. The limitations highlighted here can only be answered with careful empirical validation.

6 Summary and Conclusions

The negotiation constellations framework aims to contribute to more effective requirements engineering by capturing and structuring tactical and methodological advice that is tailored to the organizational context of a stakeholder. The framework

can be used for reflecting on the negotiation constellation, identifying other stakeholders, and obtaining guidelines for reaching agreements that increase the value of the software being developed. It may also be used for process development by providing a structure for organizing tactics and methods and to capture experience.

The negotiation constellations framework builds on the tradition of reference models like CMMI to support tactical decision-making and method selection in the area of requirements negotiation. In this role, it can help to make essential knowledge from the field of negotiation accessible to requirements engineers and software professionals and to give insights into current requirements engineering knowledge.

The paper presents and exemplifies the structure and use of the negotiation constellations framework in practical situations. It further provides specific references to tactical and methodological knowledge that can be used as a starting point for software professionals that want to address negotiation systematically and for companies that decide to adopt the framework as part of their process improvement.

Future work should cover empirical studies of how the framework is used and evolved, and of what its effects are on software quality and on learning software organizations. One aspect of interest is the evolution of the stakeholder network that emerges as a result from following a strategy built on the tactics proposed by the framework. Evaluation and comparison of requirements engineering methods from the perspective of the described negotiation constellations will make these methods better accessible to practitioners and further supports method selection.

References

1. Alexander, I., Robertson, S.: Understanding Project Sociology by Modeling Stakeholders. *IEEE Software* 21(1), 23–27 (2004)
2. Boehm, B., Grünbacher, P., Briggs, R.: Developing Groupware for Requirements Negotiation: Lessons Learned. *IEEE Software* 18(3), 46–55 (2001)
3. Cechich, A., Piattini, M., Vallecillo, A.: Component-Based Software Quality. LNCS, vol. 2693, pp. 99–127. Springer, Heidelberg (2003)
4. Cooper, R.: Winning at New Products: Accelerating the Process from Idea to Launch. Perseus Publishing (2001)
5. Curtis, B., Krasner, H., Iscoe, N.: A Field Study of the Software Design Process for Large Systems. *Communications of the ACM* 31(11), 1268–1287 (1988)
6. Damian, D., Eberlein, A., Woodward, B., Shaw, M., Gaines, B.: An Empirical Study of Facilitation of Computer-mediated Distributed Requirements Negotiations. In: 5th Intl. Symposium on Requirements Engineering (2001)
7. Doerr, J., Paech, B., Koehler, M.: Requirements Engineering Process Improvement Based on an Information Model. In: 12th IEEE Intl. Requirements Engineering Conference (2004)
8. Foa, U., Foa, E.: Resource Theory of Social Exchange. General Learning Press (1975)
9. Fricker, S., Gorschek, T., Myllyperkiö, P.: Handshaking between Software Projects and Stakeholders Using Implementation Proposals. In: 13th Intl. Working Conference on Requirements Engineering: Foundation for Software Quality (2007)
10. Glinz, M., Wieringa, R. (eds.): IEEE Software Special Issue on Stakeholders in Requirements Engineering 24(2) (2007)
11. Gorschek, T., Svahnberg, M.: Requirements Engineering in Practice: Studies of Six Companies. In: Aurum, A., Wohlin, C. (eds.) *Engineering and Managing Software Requirements*, pp. 405–426. Springer, Heidelberg (2005)

12. Grünbacher, P., Seyff, N.: Requirements Negotiation. In: Aurum, A., Wohlin, C. (eds.) *Engineering and Managing Software Requirements*, pp. 143–162. Springer, Heidelberg (2005)
13. Humphrey, W., Snyder, T., Willis, R.: Software Process Improvement at Hughes Aircraft. *IEEE Software* 8(4), 11–23 (1991)
14. Kotonya, G., Sommerville, I.: Requirements Engineering with Viewpoints. *Software Engineering Journal* 11(1), 5–18 (1996)
15. van Lamsweerde, A., Darimont, R., Letier, E.: Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Transactions on Software Engineering* 24(11), 908–926 (1998)
16. Lewicki, R., Barry, B., Saunders, D.: *Essentials of Negotiation*. McGraw-Hill, New York (2007)
17. Lumsdaine, E., Lumsdaine, M.: Creative Problem Solving. *IEEE Potentials* 13(5), 4–9 (1994)
18. Marquardt, K.: Patterns for Plug-ins. In: Manolescu, D., Voelter, M., Noble, J. (eds.) *Pattern Languages of Program Design 5*, pp. 301–336. Addison-Wesley, Reading (2006)
19. Ovaska, P., Rossi, M., Smolander, K.: Filtering, Negotiating and Shifting in the Understanding of Information System Requirements. *Scand. J. of Information Systems* 17(1), 31–66 (2005)
20. Pohl, K., Böckle, G., van der Linden, F.: *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, Heidelberg (2005)
21. Porter, M.: *Competitive Advantage: Creating and Sustaining Superior Performance*. The Free Press (1985)
22. Poston, R.: Software Standards. *IEEE Software* 1(2), 87–94 (1984)
23. Rahim, M.: *Rahim Organizational Conflict Inventories: Professional Manual*. Consulting Psychologists Press (1990)
24. Raiffa, H., Richardson, J., Metcalfe, D.: *Negotiation Analysis: The Science and Art of Collaborative Decision Making*. The Belknap Press of Harvard University Press (2007)
25. 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, Heidelberg (2005)
26. Robinson, W., Volkov, S.: Supporting the Negotiation Life Cycle. *Communications of ACM* 41(5), 95–102 (1998)
27. Rubin, J., Pruitt, D., Kim, S.: *Social Conflict: Escalation, Stalemate and Settlement*. McGraw-Hill, New York (1994)
28. Schneider, K., von Hunnius, J., Basili, V.: Experience in Implementing a Learning Software Organization. *IEEE Software* 19(3), 46–49 (2002)
29. Sommerville, I., Sawyer, P.: *Requirements Engineering: A Good Practice Guide*. Wiley, Chichester (1997)
30. Thomas, K.: Conflict and Conflict Management. In: Dunette, M. (ed.) *Handbook of Industrial and Organizational Psychology*, pp. 889–935. Rand McNally College Publishing Company (1976)
31. Thompson, L.: *The Mind and Heart of the Negotiator*. Pearson Prentice Hall, London (2005)
32. van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L.: Towards a Reference Framework for Software Product Management. In: 14th IEEE Intl. Requirements Engineering Conference (2006)
33. Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In: 3rd IEEE Intl. Symposium on Requirements Engineering (1997)

DESCRY: A Method for Evaluating Decision-Supporting Capabilities of Requirements Engineering Tools

Beatrice Alenljung and Anne Persson

School of Humanities and Informatics, University of Skövde, P.O. Box 408, SE-541 28
Skövde, Sweden
{beatrice.alenljung, anne.persson}@his.se

Abstract. Complex decision-making is a prominent aspect of requirements engineering (RE) and the need for improved decision support for RE decision-makers has been identified by a number of authors in the research literature. Decision-supporting features and qualities can be integrated in RE tools. Thus, there is a need to evaluate the decision-supporting capabilities of RE tools. In this paper, we introduce a summative, criteria-based evaluation method termed DESCRY, which purpose is to investigate to what extent RE tools have decision-supporting capabilities. The criteria and their related questions are empirically as well as theoretically grounded.

1 Introduction

RE has been recognized as being largely a decision-making process [1]. Stakeholders' decisions about the quality and functionality of a system are expressed in requirements. Other important decisions in RE concern issues such as organization, staffing, and planning. Thus, poor decisions can cause RE to fail [2]. By addressing improvement to decision-making in RE, the probability of successful systems engineering increases [1]. The RE decision-maker's abilities and capabilities can be enhanced if appropriate RE decision support is provided, e.g., through integrating decision-supporting features and qualities in RE tools. RE decision support should strive to augment the decision-making capacity of the human decision-maker [3].

To develop support for RE decision-making is, hence, a major issue for RE research [2]. However, research into the field of RE decision-making and RE decision support is still in its infancy [3].

This paper addresses evaluation of RE tools from a decision support perspective. Several evaluation methods for tool selection have been proposed, e.g. COSTUME [4], the R-TEA approach [5], and the value-based tool selection approach [6]. However, to the best of our knowledge, there is no dedicated method enabling systematic evaluation of the decision-supporting capabilities of RE tools. Our research contributes to filling this void by suggesting a summative, criteria-based evaluation method termed DESCRY¹ (Decision-Supporting Capabilities of RE tools).

¹ Apart from being the acronym for our evaluation method, it is also an English word, which means see a long way away or catch sight of.

The purpose of the method is *to investigate to what extent RE tools have decision-supporting capabilities*. This means that the method is summative. DESCRY has a user-centered perspective, which implies that the evaluator should take the RE decision-makers' perspective and estimate whether or not an RE decision-maker can perceive the existence of decision-supporting features as well as understand how to use them.

The remainder of the paper is organised as follows. Section 2 describes the research process. Section 3 introduces DESCRY. We give some concluding remarks in Section 4.

2 Research Process

The research process consists of four stages: a) literature analysis, b) case study, c) synthesis, and d) method development. The literature analysis resulted in a generic decision situation framework [7], which was used during stage b) to make sure that all the fundamental aspects were taken into account. The case study was conducted using a qualitative research approach. The case study took place at a systems engineering company that develops highly advanced systems. The data collection techniques were open-ended interviews and a focus group session. The interviewees were requirements engineers and stakeholders related to them. Seventeen persons participated in the study. The result from the case study was a portrayal of the decision situation of RE decision-makers [8], [9], [10]. In the synthesis, the empirical findings from the case study were synthesised with existing relevant theories. This resulted in empirically based desirable high-level characteristics of an envisioned future RE decision support system (REDSS) and guiding principles for designing such a system that are empirically as well as theoretically grounded [9]. In the method development stage, DESCRY was developed. The criteria in the method are based on the characteristics and guiding principles of REDSS. To obtain indications of the method's usefulness, we applied it to an existing RE tool, CaliberRM.

3 DESCRY – Evaluation Method of Decision-Supporting Capabilities of RE Tools

The evaluator should assess the RE tool in relation to criteria. The criteria are exclusively derived from empirical findings [9]. They are based on the needs of RE decision-makers and the nature of RE decision-making, i.e. we have focused on what is generic, and not on specific RE tasks. For each criterion, there are one, two, or three evaluation questions to facilitate the evaluation. The guiding principles are empirically and theoretically grounded. For each question, we give some additional queries in order to provide examples of how the questions can be interpreted. Extensive descriptions of the criteria are presented in [9].

The evaluator should explore the RE tool in order to evaluate it. The guiding star when answering the evaluation questions and the additional queries is: What is the likelihood that the RE decision-maker can perceive this affordance? This means that the evaluator should not just identify if there are features that can fulfill a criterion. The evaluator should also take the RE decision-maker's perspective and estimate

whether or not he or she can perceive the existence of the features and understand how to use them. Hidden or cumbersome features will most likely not improve the RE decision performance and will probably not satisfy the RE decision-makers.

Criteria 1: Reduce the cognitive load

- Is it possible to obtain both overview and details?
 - Can the RE decision-makers see the information details in a relevant context so that the understanding and use of the details are facilitated?
- Is memory aid provided?
 - Is it possible for the RE decision-maker to get or activate alerts; write and retrieve rationale for decisions; write and retrieve “soft” information (e.g., personal experiences, rumors, and opinions of others).

Criteria 2: Ensure high usability

- Are usability design principles followed?
 - Are available functions and the status of the RE tool visible?
 - Are design features used in a consistent way?
 - Are the terminology and symbols familiar to the RE decision-makers?
 - Can the RE decision-maker perceive how to use the functions?
 - Can the RE decision-maker easily navigate in the system?
 - Are there clear and logical mappings between controls and effects?
 - Is feedback constantly and consistently provided?
 - Are slips and mistakes rapidly and effectively recovered?
 - Are there constraints that prevent inappropriate actions?
 - Can the RE tool be used in a flexible way and is it possible for the RE decision-maker to personalize it?

Criteria 3: Support availability of different types of information

- Is the information mentally available, in terms of being visualized and easy to understand?
 - Is the information visualized appropriately in relation to how it should be used?
 - Is it possible to access information from outside the immediate environment of the RE tool?
 - Are there visual knowledge tools for pattern detection and knowledge crystallization?
 - Is it possible to visually enhance objects?
- Is information in different formats available?
 - Is it possible to manage data in a database?
 - Is it possible to gather, retrieve, classify, and manage unstructured documents?

Criteria 4: Support different types of decision matters

- Are decisions concerning requirements as such supported?
 - Are system-related requirements decisions, e.g., requirements prioritization, facilitated?

- Are decisions of determining suitable ways to carry out the RE process supported?
 - Are work-related RE decisions, e.g., choosing requirements acquisitions method, facilitated?
- Are decisions that are made in other parts of systems engineering that use requirements as input supported?
 - Are requirements-related decisions, i.e., beyond requirements decisions and RE decisions, e.g., test case selection, facilitated?

Criteria 5: Support creativity and idea generation

- Are techniques that enhance creativity available?
 - Is it possible for the RE decision-maker to be exposed to creativity enhancing information?
 - Are brainstorming activities supported?
 - Is idea generation in groups supported?

Criteria 6: Support knowledge sharing and transfer

- Are there ways to share and transfer knowledge?
 - Is knowledge of the application domain shared and transferred?
 - Is knowledge of RE practice shared and transferred?
 - Is knowledge of the developed system/system to be shared and transferred?

Criteria 7: Support idea evaluation and problem solving

- Are evaluation techniques available?
 - Is it possible to quantitatively compare the effectiveness of suggested alternatives?
 - Is it possible to find out what will happen to a suggested solution if some aspect changes, e.g., an input variable, an assumption, or a parameter value?
 - Is it possible to calculate which values of the input are required in order to accomplish a preferred level of a goal?
- Is it possible to externalize a problem representation?
 - Is it possible for the RE decision-maker to draw and make use of concept maps of problems, i.e., graphical representations in which concepts are linked to other concepts?

Criteria 8: Support decision communication

- Are additional communication paths provided?
 - Is it possible to communicate with decision stakeholders via the RE tool, e.g., via chat systems, interactive whiteboards, bulletin boards, shared information spaces, or virtual meeting systems?
 - Is it possible to disseminate decisions to stakeholders?
- Are negotiation facilities provided?
 - Does the RE tool support bargaining, consensus seeking, or conflict resolution?

Criteria 9: Support coordination

- Are coordination technologies available?
 - Is it possible to manage interdependencies between activities to harmonize them?
 - Is it possible to specify behaviours of the human actors, e.g., by establishing shared goals?
 - Is it possible to plan behaviors of the human actors, e.g., by agreeing the set and order of tasks?
 - Is it possible to schedule behaviours of the human actors by, e.g., assigning tasks to individuals or groups?

4 Concluding Remarks

In this paper, we suggest a summative, criteria-based evaluation method called DESCRY. The purpose of DESCRY is to find out to what extent RE tools have decision-supporting capabilities. The criteria are empirically grounded. The related questions are empirically as well as theoretically grounded.

DESCRY is intended to be used by practitioners as well as researcher. RE tool buyers can use it in comparing available tools in order to assess which one provide appropriate decision-supporting capabilities. RE tool developers can use DESCRY to identify the potential of a certain improvement. In addition, DESCRY is intended to serve as a road map that can direct efforts of researchers addressing RE decision-making and RE decision support problems. Our intent is to widen the scope and give new lines of thought about how decision-making in RE can be supported and improved.

However, the usefulness of DESCRY is not yet validated. So far, it has not been used by others than its inventors and its actual usefulness cannot be concluded without thorough evaluation involving its intended users. This clearly requires further research. In addition, DESCRY is, as mentioned before, summative, which means that it is concerned with the intrinsic values of the evaluation object, i.e. the RE tools. Hence, its purpose is not to suggest changes of the tools. Future research can be directed to transform the current summative evaluation method into a formative method. Such a method would make it easier for an RE tool developer, who intend to increase the decision-supporting capabilities of a tool, to obtain concrete ideas for improvements.

References

1. Aurum, A., Wohlin, C.: The fundamental nature of requirements engineering activities as a decision-making process. *Information and Software Technology* 45, 945–954 (2003)
2. Regnell, B., Paech, B., Aurum, C., Wohlin, C., Dutoit, A., Nattoch Dag, J.: Requirements means decision! – Research issues for understanding and supporting decision making in requirements engineering. In: 1st Swedish Conference on Software Engineering Research and Practice, SERP 2001, Ronneby, Sweden (2001)

3. Ngo-The, A., Ruhe, G.: Decision support in requirements engineering. In: Aurum, A., Wohlin, C. (eds.) *Engineering and managing software requirements*, pp. 267–286. Springer, Berlin Germany (2005)
4. Carvallo, J.P., Frach, X., Quer, C.: A quality model for requirement management tools. In: Maté, J.L., Silva, A. (eds.) *Requirements engineering form sociotechnical systems*, pp. 119–137. Information Science Publishing, Hershey (2005)
5. Matulevičius, R.: Comparing goal-modelling tools with the RE-tool evaluation approach. *Information Technology and Control*, 276–286 (2006)
6. Heindl, M., Reinisch, F., Biffl, S., Egyed, A.: Value-based selection of requirements engineering tool support. In: *Proceeding of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications, EUROMICRO-SEAA 2006* (2006)
7. Alenljung, B., Persson, A.: Decision-making from the decision-maker's perspective: A framework for analysing decision situations. In: *Proceedings of the 4th International Conference on Business Informatics Research, BIR 2005*, Skövde, Sweden, October 3-4 (2005)
8. Alenljung, B., Persson, A.: Decision-making activities in the requirements engineering decision processes: A case study. In: *Proceedings of the 14th International Conference on Information Systems Development, ISD 2005*, Karlstad, Sweden, August 15-17 (2005)
9. Alenljung, B.: *Envisioning a future decision support system for requirements engineering: A holistic and human-centred perspective*. Doctoral Thesis, Department of Computer and Information Science, Linköping University, Sweden, Thesis No. 1155 (2008)
10. Alenljung, B., Persson, A.: Factors that affect requirements engineers in their decision situations: A case study. In: *Proceedings of the 11th International Workshop on Requirements Engineering: Foundation for Software Quality, REFSQ 2005*, Porto, Portugal, June 13-14 (2005)

Inventing Requirements: Experiences with an Airport Operations System

Neil Maiden¹, Cornelius Ncube², and James Lockerbie¹

¹ Centre for HCI Design, City University London, UK
N.A.M.Maiden@city.ac.uk, J.Lockerbie@soi.city.ac.uk

² Software Systems Research Centre, Bournemouth University, UK
cncube@bournemouth.ac.uk

Abstract. This paper reports a workshop that integrated creativity techniques with extended use case diagrams and storyboard representations of use cases to discover stakeholder requirements for VANTAGE, a new system designed to reduce environmental impact at airports. The workshop revised the boundaries of the system and generated 200 new requirements-based ideas and storyboards for VANTAGE. The paper describes the workshop structure, gives examples of outputs from it, and uses these outputs to answer 3 research questions about the usefulness of ideas generated and creativity techniques employed.

1 Introduction

As we have reported previously [1, 2], requirements engineering is a creative process in which stakeholders and engineers work together to create ideas for new software systems that are eventually expressed as requirements. The importance of creative system and product design is increasing. Creativity is indispensable for more innovative product development [3], and requirements are recognized as a key abstraction that encapsulates the results of creative thinking about a system.

Most current requirements processes and research activities support problem analysis and system specification. In contrast, invention is often perceived as part of the design process that follows requirements engineering [4]. One assumption behind research approaches such as *i** [5] and commercial processes such as the RUP is that stakeholders have sufficient knowledge to already know their requirements. However, this is increasingly flawed because of the breadth of expertise that is needed to specify complex systems and the need for stakeholders with different areas of expertise to work together to generate requirements.

One challenge is to build on previous successes [6, 7, 8] and integrate creativity techniques into mainstream requirements engineering processes. This paper reports unpublished results from one creativity workshop within the RESCUE requirements process [8] that was run to discover requirements for a new system to reduce the environmental impact of ground aircraft movements at airports. The two-year VANTAGE (*Validation of a Network-Centric, Technology Rich ATM System Guided by the Need for Environmental Governance*) Phase-1 project, funded by the UK's Department of Trade and Industry, integrates technologies into the operations of

regional airports in the United Kingdom to reduce their environmental impact, measured as noise and gas emissions. Partners who included Thales and Qinetiq were introducing new technologies such as surveillance systems into airport operations at Belfast City Airport (BCA) in Northern Ireland, the pilot site for the project. The VANTAGE requirements process sought to determine new requirements and opportunities arising from the technology-led changes to the complex socio-technical systems at BCA, and in particular to the work practices of actors such as air traffic controllers, dispatchers and refueling staff. Requirements challenges specific to VANTAGE included exploring the complex boundaries of airport operations, determining the impacts on work practices that might be changed, and specifying new interactive mobile tools that airport staff might use to reduce the environmental impact of aircraft being turned around at BCA.

In the applied requirements process we ran one creativity workshop to explore the boundaries of airport operations at BCA, discover new requirements and design features on interactive and other technologies to be installed at the airport, specify changes in work practices at BCA, and generate first-cut use case specifications of VANTAGE that informed later requirements processes in RESCUE. We used outputs from the workshop to answer 3 research questions about the usefulness of the requirements and design features generated from applying individual and combined creativity techniques, and the effectiveness of extended requirements modeling notations to support creative thinking about system boundaries. Results have implications for the redesign of requirements processes, techniques and notations to support more effective creative thinking at the start of systems development.

2 RESCUE and Its Creativity Workshops

RESCUE is a concurrent engineering process in which different modeling and analysis processes take place in parallel [9]. Concurrent processes are structured into 4 streams. The two most important streams are:

1. System goal modeling to model the future system boundaries, actor dependencies and most important system goals;
2. Use case modeling and scenario walkthroughs to communicate more effectively with stakeholders and acquire complete and testable requirements.

Creativity workshops normally take place after a requirements team has specified the system boundaries using context and use case diagrams but before it specifies the detailed use cases. Their main purpose is to discover and invent requirements and ideas needed to specify use cases.

We designed RESCUE to separate the creativity workshops from other more practical requirements activities such as use case specification, requirements acquisition and requirements management. In the VANTAGE project, the requirements team undertook these other requirements activities before and after the workshop.

2.1 Previous Creativity Work

As we have reported previously [6], little requirements engineering research has addressed creative thinking directly. Brainstorming techniques and RAD/JAD

workshops [10] make tangential reference to creative thinking. Most current brainstorming work refers back to Osborn's text [11] on principles and procedures of creative problem solving (CPS). However, there are no reported applications of the CPS model to requirements processes. Robertson [12] argues that requirements analysts need to be inventors to bring about innovative change that gives competitive advantage. Nguyen et al. [13] observed that teams restructured requirements models at critical points when they solve sub-problems, triggered by moments of sudden insight. Mich et al. [14] report the successful use of the elementary pragmatic model from communication theory to trigger combinatorial creativity during requirements acquisition. However, none of these approaches exploit creativity theories or models directly. Requirements analysts still lack processes to guide their creative processes.

2.2 Creativity Workshops in RESCUE

RESCUE incorporates creativity workshops to encourage creative thinking with which to invent requirements. As we reported previously [6], the workshop activities are designed using 3 established models of creativity from cognitive and social psychology that we use for 3 purposes. Firstly, to encourage creative thinking, it is essential to establish a working definition of creativity. The models provide us with such a definition. Secondly, it is important to structure the workshops into creative processes. The models provide us with taxonomies of creative thinking with which to structure processes in workshops. Thirdly, one model provides procedural guidance for creative problem solving that we apply directly to each workshop's design.

In RESCUE we adopt Sternberg's [15] definition as prototypical of those available in the literature. Creativity is defined as "*the ability to produce work that is both novel (i.e. original, unexpected) and appropriate (i.e. useful, adaptive concerning task constraints)*". As with previous projects we designed the VANTAGE creativity workshop to produce ideas that were novel in the VANTAGE domain, novel to VANTAGE stakeholders, and useful for VANTAGE according to these stakeholders.

So how did we apply the 3 creativity models? Firstly, we designed the workshop to support the divergence from and convergence towards ideas as described in the CPS model [11]. As such each workshop period, which typically lasted half a day, started from an agreed current system model, diverged, then converged towards a revised agreed model that incorporated new ideas at the end of the session. Secondly, we designed each workshop period to encourage one of 3 basic types of creativity identified by Boden [16] – exploratory, combinatorial and transformational creativity. These 3 types are based on computational creativity approaches that define a space, then explore and transform it. Thirdly, we designed each period to encourage 4 essential creative processes reported in [17]: preparation, incubation, illumination and verification. Poincare's philosophical model was based on personal reflections about his own scientific processes. We designed incubation and illumination activities using the type of creativity that we sought to encourage.

In RESCUE we did not integrate these 3 creativity models directly in a single, consistent model of requirements creativity. Rather these models contributed separately to the design a coordinated creative requirements process. The CPS model processes provided the overall structure of the process. During each period the process encourages divergence from a current requirements model then convergence

towards a new one. Poincaré's model provided finer-grain processes – incubation and illumination – to achieve this divergence and convergence. Boden's types of creativity were used to select different creativity techniques for achieving incubation and illumination during convergence and divergence. It is these techniques that are the focus of the reported results. A 2-day workshop is composed of 4 half-day creativity periods. In each period we use a different creativity technique to encourage different types of creativity.

Prior to VANTAGE, the RESCUE team had facilitated 10 creativity workshops in the air traffic and policing domains that were reported in previous publications [1, 6, 7, 8, 18]. However, project pressures and the absence of available resources meant that we had been unable to explore the impact of creativity workshop ideas on requirements specifications. In VANTAGE, resource and time was put aside for key stakeholders to assess creativity workshop outputs and, through them, the effectiveness of the workshops to generate requirements that can be implemented in VANTAGE.

3 The VANTAGE Creativity Workshop

The VANTAGE creativity workshop took place on the 19th and 20th April 2006 at Belfast City Airport in Northern Ireland. The workshop ran 10.00–17.00hrs on the first day and 09.00–16.30hrs on the second. It involved 4 creativity sessions: (i) brainstorming, then challenging boundaries; (ii) exploring constraints; (iii) discovering requirements from solutions; (iv) storyboarding. In each session a different creativity technique was used to encourage different types of creativity.

One facilitator, 2 scribes, and 11 participants attended the VANTAGE workshop. Each participant represented a VANTAGE technology partner, the national air traffic service, the airport, the community forum or the airport operators association. The workshop was held in a large meeting room. The use case models and précis provided the structure for the workshop room. Each diagram and précis was posted on a separate 1m² pin board in the workshop room that became the physical and logical structure of ideas associated with use cases during the workshop. In total there were 27 such use cases and pin boards at the start of the workshop.

On day-1 in the morning session the participants brainstormed new ideas for VANTAGE, then walked through the VANTAGE use case diagram shown in Figure 1 to review the VANTAGE system boundaries. Analysts had developed the diagram prior to the workshop, and a larger version of it is available at [22]. In the afternoon session participants brainstormed constraints on VANTAGE, then removed selected constraints to generate new requirements and design features for VANTAGE. On day-2 in the morning, the technology partners presented technologies available for use in VANTAGE. Afterwards all of the participants generated new requirements and design features based on these available solutions and combined them with outcomes from day-1. In the afternoon participants developed 3 storyboards to combine ideas from the first one and half days. The use cases were prioritized, then 3 groups took the 3 highest priority use cases and constructed storyboards for them. Furthermore, participants were encouraged to generate new requirements and design features that surfaced as a result of the workshop activities and document them on ideas cards that were placed on the

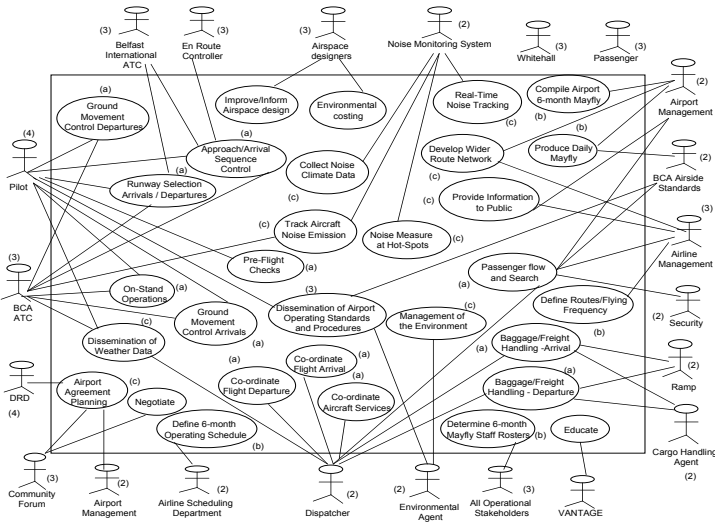


Fig. 1. The use case diagram at the end of the workshop

relevant pin boards. To do this the participants were supplied with A6 RESCUE colour-coded idea cards, post-it notes, A3 paper and pens. Everything captured on the pin boards was documented electronically in a workshop report sent to all VANTAGE stakeholders at the end of the workshop.

These activities and the 4 sessions are described in more detail.

3.1 Exploratory Creativity

The exploratory creativity session was in two parts – brainstorming then scoping. During the brainstorming part standard RAD/JAD facilitation techniques and rules [19] such as avoiding criticism of other people's ideas and time-boxing each topic under discussion were applied. Participants reported requirements during round-robin then open-ended brainstorming activities.

During the scoping part participants worked together to identify to what extent VANTAGE can redesign the work of each actor modeled in the use case diagram. We encouraged exploratory creativity by challenging VANTAGE system boundaries and exploring ideas previously outside VANTAGE's scope. We applied an extension to the use case diagram notation to represent different system boundaries. UML use case diagrams have a simple representation that describes actors outside of a system boundary, rather than the complex types of actors and boundaries found with socio-technical systems such as VANTAGE. Therefore each actor was identified as either: (1) a new system introduced by VANTAGE; (2) human work to be redesigned in VANTAGE; (3) systems and work outside of VANTAGE's direct redesign but open to influence by it, and; (4) systems and work outside of VANTAGE's scope. Figure 1 shows these actor system boundaries, for example the *cargo handling agent* is tagged

with (2) – work to be redesigned by VANTAGE partners – and the *en-route controller* is tagged with (3) – cannot be redesigned directly by VANTAGE partners, but amenable to influence by solutions that VANTAGE will deliver. We chose to use simple tags to make the use case diagram simple to change. The viscosity of a notation is a cognitive dimension [20] often found in requirements methods. Whilst less important during more formal analyses, viscosity can be an impediment to creative requirements modeling, and we sought to avoid it in the creativity workshop.

The use case diagram was also tagged to indicate the impacts of the use cases if implemented successfully – a form of prioritization in VANTAGE. Participants ranked each use case as having: (a) a direct potential benefit for the environment; (b) a partial impact or (c) no impact. Again the use case diagram was extended using a simple notation to indicate each potential impact. Figure 1 shows that the *ground movement control departures* use case was tagged (a) – direct potential impact – whilst the *develop wider route network* use case was tagged (c) – having no impact.

An initial version of the use case diagram, with tagged actors and use cases, was developed prior to the workshop, although the 27 use cases were reduced to 22 after an initial review of the diagram at the beginning of the workshop.

Finally, during the day-1 lunch period, the facilitator asked participants to think of other worlds and systems familiar to them, and generate new VANTAGE ideas based on analogical mappings with these worlds and systems.

3.2 Transformational Creativity

During transformational creativity people change the solution space in a way that things that were considered impossible are now possible. On the afternoon of day-1 we encouraged transformational creativity by guiding participants to discover and remove constraints on the reduction of the environmental impact and air travel at BCA. The facilitator led a group brainstorming session to discover as many constraints as possible. Participants then worked in 3 groups to select constraints in turn until none remained, then envisaged the removal of each constraint to generate new VANTAGE ideas based on this removal. The session ended with the groups reporting new VANTAGE ideas and posting them on the ideas boards, which in turn led to a final period of group brainstorming using the new ideas.

On the morning of day-2 five participants gave 5-10 minute presentations of candidate VANTAGE technologies to discover new requirements and opportunities. These technologies were collaborative data networks from Selex SI, ADS-B air-derived surveillance from Raytheon Systems, an interactive approach path monitor from Thales S.A., enhanced interactive display equipment from Flight Refueling, and an airport synthetic environment from QinetiQ. After these presentations, participants worked in groups to discover new requirements and design features arising from these solutions, which were again documented on ideas cards and reported back to the other groups at the end of the session.

3.3 Combinatorial Creativity

Combinatorial creativity is the creation of new ideas from combination and synthesis of existing ideas. It is the act resulting from an unusual combination of existing

concepts [16]. At the end of the morning of day-2 participants combined ideas generated from the solution presentations with ideas on the pin boards generated from earlier techniques. On the afternoon of day-2, storyboarding was used to elaborate and combine creative ideas in the last period of the workshop. Participants again worked in 3 groups. Each group was asked to produce a storyboard that described the possible combination of ideas associated with one use case during the first 3 periods of the workshop. To structure the storyboarding process, each group was given A1-size pieces of paper that were annotated with 16 boxes to contain a graphical depiction of each scene of the storyboard and a space to describe that scene.

3.4 So How Useful Were the Creativity Workshop Ideas?

We used data gathered during and after the workshop to investigate 3 research questions about the usefulness of the generated requirements and design features in the VANTAGE project, and the extended UML use case diagram notations to support creative thinking about system boundaries. Although creativity workshops in earlier projects generated ideas that were perceived to be novel by stakeholders, concerns remained about the usefulness of these ideas in subsequent requirements and development processes [8]. Therefore we used the VANTAGE workshop results to answer 2 research questions about the usefulness of generated requirements and design features, and thereby and the techniques that led to their generation:

- Q1 Did the creativity workshop, as an event, generate ideas that were perceived to be useful when specifying the requirements of the system to be implemented?
- Q2 Did individual creativity techniques generate ideas that were perceived to be useful when specifying the requirements of the system to be implemented?

We also used the VANTAGE workshop results to answer a third question about the usefulness of the extended the UML use case diagram notation. Existing requirements notations such as i^* [5] and the UML were not designed to support creative thinking. Indeed requirements notations are often designed for analysis rather than invention, and in cognitive terms can be described as viscous – resistant to change. Therefore we sought to answer a third research question:

- Q3 Did the extended UML use case diagram support boundary changes?

We used ideas – requirements and design features – and model changes generated during the workshop, post-workshop ranking of ideas by VANTAGE stakeholders and our own observations of the workshop to seek answers to these 3 research questions.

4 Results

The workshop took place and ran to schedule, and all activities were followed without disruption. Minor conflicts about ideas were handled with facilitated discussion during the report back presentations and verification activities.

The main outcomes are summarized in Table 1. Overall the participants generated 197 ideas and 3 storyboards over the 2 days. Participants produced 34 new

Table 1. Totals of ideas generated by technique, for the VANTAGE system and specific use cases

Deliverable type	Number system-wide	Number use case-specific
Brainstormed ideas	18	16
VANTAGE constraints	31	0
Ideas from VANTAGE constraints	113	0
Ideas from VANTAGE solution presentations	0	6
Workshop1 storyboards	0	3 storyboards
Ideas from informal analogies	3	2
Non -Technique Specific Ideas	39	0

VANTAGE ideas from the day-1 brainstorming and scoping session, another 113 by identifying and removing 31 constraints on VANTAGE, 6 ideas from presentations of VANTAGE technologies, 5 ideas from the informal lunchtime analogies, 3 large storyboards and 39 ideas placed on pin boards but not ascribed to a specific technique.

Most ideas were attributed to the VANTAGE system rather than specific use cases. Exceptions to this were the brainstormed ideas – almost half were attributed to use cases, the 6 ideas from the solution presentations all attributed to use cases, and the 3 storyboards that were generated for selected use cases. This contrasts with previous RESCUE creativity workshops [e.g. 6], in which more than half of all generated ideas were attributed to use cases.

4.1 Exploring System Boundaries

On the morning of day-1 we reviewed the use case diagrams with information about the redesign of actor work and the environmental impact of behaviour expressed in use cases shown in Figure 1. After the workshop we investigated the changes recorded on the use case diagram during this period, see Table 2. During the 90-minute session the participants recorded 12 changes to the actors on top of the 34 new ideas reported on the ideas cards. Four new actors were added to the diagram, 3 actors already on the diagram but outside VANTAGE boundaries were moved inside VANTAGE boundaries, and a further 5 actors already inside VANTAGE boundaries were changed.

The addition of the 4 new actors demonstrates the benefits of reviewing the use case diagram during brainstorming sessions. Even though the diagram had undergone thorough analyses prior to the workshop, walking through it still revealed missing actors, such as *passengers*, *air space designers* and *UK Government*, which were to

Table 2. Changes to use case diagram during exploratory creativity

Change to use case diagram	Number	Actors changed
New actors added to the model	4	Whitehall (UK Government); Passengers; Airspace designers; VANTAGE
System boundaries extended to include actors	3	Environmental agent; Community forum; DRD
System boundaries modified to change actor roles	5	Belfast International ATC; En-route controller; Noise monitoring system; operational stakeholders; BCA ATCOs

be influenced by the introduction of the VANTAGE system. Of the 34 ideas generated during the brainstorming session 4 made reference to passengers (e.g. *better and improved idea who stakeholders are and better communication with public*), suggesting some thematic overlap between brainstormed ideas and changes to the diagram.

Tagging actors and use cases in the diagram was intended to support creative exploration of VANTAGE boundaries and include more actors in the system. Results indicate that it occurred as planned. Two actors - *community forum* and *DRD (Department of Regional Development)* - were brought into the scope of VANTAGE as a result whilst a third - *environmental agent* - was changed to redesign the agent's work.

More surprisingly, 5 actors were taken outside of the scope of the VANTAGE direct work redesign. *Air traffic controllers at Belfast International Airport, en-route controllers, BCA controllers and other operational stakeholders* were changed to have their work influenced rather than redesigned by VANTAGE. The simple extension to use case diagram notations fundamentally changed the boundaries of the complex socio-technical system in 1 hour of work. Again, of the 34 ideas generated, some referred to air traffic controllers (e.g. *system to give extra information to controllers that is useful but does not overload them – incentivize them*), suggesting more thematic overlap between the brainstormed ideas and the diagram changes.

4.2 Constraint Generation Works

Identifying and removing VANTAGE constraints to discover new ideas took place on the afternoon of day-1. Participants worked together to discover 31 constraints on the design of VANTAGE. These constraints were then divided between 3 groups, each containing 3 or 4 participants. The groups worked in parallel to brainstorm 113 VANTAGE ideas that became possible if a selected constraint was removed. A final report back session communicated the ideas across the 3 groups.

Removing constraints led participants to generate new VANTAGE ideas. For example removing constraint [C17] *variability of weather* quickly led to generation of 5 ideas: (i) *capacity – use cross wind (predictable) to blow away vortex*; (ii) *gliding approaches and steam catapult departures – no power*; (iii) *always choose runway that would give best environmental impact*; (iv) *more certainty for people around airports in terms of what noise they will get when*; (v) *round runways, steam catapult runways*. These ideas demonstrate how removal of a single constraint often generated some ideas that both could be implemented in VANTAGE (e.g. *use of predictable crosswinds*) and some that could not (e.g. *steam catapults*). We investigated this phenomenon more systematically using the post-workshop data reported in Section 5.

Similarly removing constraint [C21] *different airline operating methods* led to the generation of 3 ideas: (i) *simpler model and consistent measurements to validate the model*; (ii) *better predictability = standardized turnaround times and operations*; (iii) *applicable to both strategic and tactical planning*. Further analysis revealed that these 3 ideas could be implemented to some degree in VANTAGE without removing the original constraint. It demonstrates the potential effectiveness of constraint removal on generation of ideas that are potentially useful in requirements processes. Again we investigated this phenomenon more systematically in Section 5.

More occasionally the removal of constraints led stakeholders to consider trade-offs between satisfying competing goals. For example, removing constraint [C18] *ability to grow with demand* led to the generation and documentation of advantages of the constraint's removal: (i) *increased traffic throughput*; (ii) *reduced emissions*; (iii) *increased throughput and revenues*; (iv) *improved gate layout*; (v) *terminal layout-taxi distance reduces with engine size*. Elsewhere, removing constraints led stakeholders to consider the possible advantages and disadvantages of VANTAGE ideas. For example, removing constraint [C3] *airport operating hours* led to advantages such as *aircraft movement increases* and *increased aircraft utilization*, but also disadvantages such as *major impact on 24-hour noise disturbance* and *increased community disturbance*. As such the stakeholders explored trade-offs between soft goals that are more commonly expressed with requirements notations such as *i** soft goal contribution links [5].

4.3 Storyboarding Use Cases

On the afternoon of day-2 the 3 stakeholder groups combined ideas from the physical use case pin boards to produce 3 storyboards for the 3 prioritized use cases using structured storyboards. Figure 2 shows the state of the use case *UC4 on-stand operations* at the end of the workshop. The original input to the workshop was a simple précis of just 12 words. The final use case storyboard depicts how wind direction changes are detected by a VANTAGE system that recommends a change of stand for an arriving aircraft and co-ordinates ground staff to that stand. It includes and combines ideas e.g. *have an accurate ETA (estimated time of arrival)*. A larger version of the storyboard is available at [21]. The other 2 storyboards were developed to a similar level of detail.

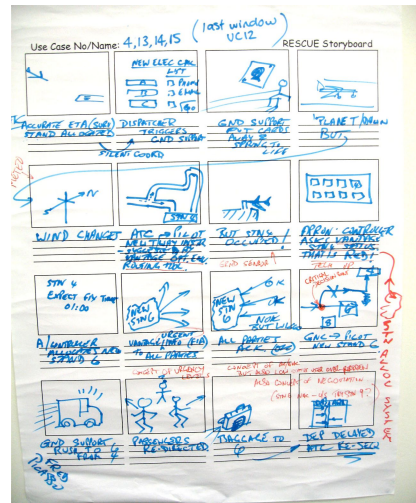


Fig. 2. The storyboard for the UC4 On-stand operations use case

5 Rating Idea Usefulness

Although participants considered the VANTAGE creativity workshop a success at the time that it took place, and the team used its outcomes in subsequent RESCUE activities, we were unable to review the perceived usefulness of the individual ideas. Therefore 3 months later, 6 volunteer VANTAGE stakeholders reviewed the usefulness of 169 of the 197 workshop ideas. All 6 reviewed all of the 169 ideas. We dropped ideas of 2 types from this analysis – the 24 ideas specific to use cases, which could only be understood by presenting the use cases as well, and 4 system-wide ideas that were self-comments to the participants rather than documented ideas per se.

For the purposes of VANTAGE, usefulness was defined as the degree to which each idea could be implemented in the VANTAGE system. A simple 3-point scale was adopted for each idea – 1 indicated that it can definitely be implemented in VANTAGE, 2 indicated that the idea might possibly be implemented in VANTAGE, and 3 indicated that the idea could not be implemented in VANTAGE. Six stakeholders who had attended the workshop – 4 from the technology partners, 1 from the airport and 1 from the community forum – independently ranked each of the 169 workshop ideas.

Results are reported in Table 3. The stakeholders, on average, ranked just over 51 ideas as definite implementations in VANTAGE, 62 as possible implementations and 55 as not to be implemented. There were differences between types of stakeholder. The technology stakeholders tended to rank fewer ideas as definite implementations – an average of just over 34 per stakeholder – and more ideas as not to be implemented – an average of almost 72 per stakeholder.

Table 3. Totals of ideas ranked as definitely, possibly and not to be implemented by the 4 technology partners T1-T4, the airport stakeholder A1 and the community forum representative C1

Idea ranking	T1	T2	T3	T4	A1	C1	Average
Definitely to be implemented	25	17	82	13	76	95	51.3
Possibly to be implemented	107	54	44	44	62	61	62
Not to be implemented	36	96	43	112	30	13	55
Not ranked	1	2	0	0	1	0	0.66

Analysis by idea also revealed different stakeholder ratings. All 6 stakeholders ranked only 3 ideas as definite implementations: (i) *better and improved idea who stakeholders are and better communication including public*; (ii) *VANTAGE is scalable – not a constraint*; (iii) *focus on noise, air quality and emissions*. One arose during the day-1 brainstorm, the second during day-1 constraint removal, and the third was not attributed to a specific technique. All 3 ideas refer to requirements specified in the original VANTAGE project plan. A further 10 ideas were ranked as definite implementations by 4 of the 6 stakeholders. Four of these ideas were generated during the day-1 brainstorm, 5 during day-1 constraint removal and 1 was again not attributed to a technique. Three of the 6 stakeholders ranked a further 37 ideas as definite implementations. Once more, a range of creativity techniques generated these 37 ideas.

Ideas generated from each of the 31 VANTAGE constraints were analyzed to determine the constraints for which both ideas that were both implementable (e.g. *use of predictable crosswinds* reported in section 4.2) and not implementable (e.g. *steam catapults*). An idea was recorded as implementable if half or more (≥ 3) of the stakeholders rated it as such, whilst an idea was recorded as not implementable if less than half or less (< 3) of the stakeholders rated it as such. Eight of the 31 constraints led to ideas that were both implementable and not implementable, suggesting that for some constraints, participants generated ideas both useful and not useful.

To summarize, the post-workshop analysis revealed that one third of the ideas, generated with different creativity techniques, were useful. The workshop ideas appeared to be more useful to some stakeholder types than others. Given the advances

made in the 2 days – revision of VANTAGE boundaries, actors and use cases, storyboarding of the most important use cases, and generation of ideas that VANTAGE can implement – we believe that the workshop was useful. However, there appear to be limits on who benefited most from the workshop and how.

6 The Research Questions Revisited

VANTAGE stakeholders regarded the creativity workshop as a success. The VANTAGE team later used deliverables from it to generate the OCU specification for the VANTAGE system. Workshop results and data gathered through the post-workshop review provided data to answer the 3 research questions.

The first question (Q1) asked whether the creativity workshop generated ideas that were useful when specifying the requirements of the system to be implemented. Results are inconclusive. Whilst, on average, almost one-third of the VANTAGE ideas were identified as useful, there were differences between stakeholders. Between them, the airport and community stakeholders rated over half of the ideas as implementable, indicating that potential beneficiaries of technologies discovered new requirements and opportunities to negotiate the satisfaction of later on. In contrast, 3 of the technology partners rated less than one-sixth of the VANTAGE ideas as useful. One possible explanation was that these requirements could not be implemented using their technologies. Therefore, for a new socio-technical system based on known technologies, the workshop was more useful to the actors in this system. However, the small number of ideas generated after the 5 technology presentations indicated that such direct presentations did little to communicate the opportunities that the technologies offered.

The second question (Q2) explored whether individual creativity techniques generated ideas that were useful when specifying requirements of the system to be implemented. Constraint removal was more productive in terms of the number of ideas generated, and brainstorming led to more ideas ranked after the workshop as definitely to be implemented. Stakeholders also generated 3 detailed and complete storyboards. In contrast, presentations of solution technologies and use of informal analogies led to low numbers of ideas. These outcomes are discussed in turn.

Transformational creativity with constraint removal generated more ideas than other any single creativity strategy. One possible reason for this was that we also counted possible advantages and disadvantages such as those reported above as separate ideas. Nonetheless, constraint removal appeared to be a productive technique for generating ideas that were potentially both useful and not in subsequent requirements processes.

Brainstorming led to the generation of smaller numbers of ideas, however post-workshop analyses revealed that these ideas were more likely to be implemented. One explanation is that brainstorming was the first technique applied, and led stakeholders to express more obvious, considered ideas already planned to be implemented.

In contrast the 90 minutes of transformational creativity using the presentations of VANTAGE technologies generated only 6 ideas, or 3% of all ideas generated during the workshop. This contrasted with earlier workshops in which presentations of information visualization solutions had led to generation of 9% of all ideas generated during one two-day workshop [1] and 7% of all ideas during 3 two-day workshops in the air traffic control domain [6].

So what was different about these previous workshops? One was the use of a book of a large number of candidate information visualization solutions to choose from, rather than the 5 mandated technical solutions in VANTAGE. In the earlier workshops, instead of simply reviewing each presented solution in turn, participants worked together to generate new information visualizations. Another difference was the style of presentation. Whereas the information visualization expert provided neutral descriptions of each visualization solution, the solution providers in VANTAGE stressed important features of their solutions to other participants, which might have impacted on subsequent idea generation.

Furthermore, 39 ideas were placed on pin boards but not ascribed to one technique. There are several possible explanations for this. We discount the possibility that the facilitator and scribes did not manage idea traceability during the workshop because these 39 ideas had stakeholders who claimed ownership of them. Rather stakeholders completed these 39 ideas during the workshop between planned creativity periods, indicating the importance of the workshop as a single entity to generate some of the requirements and design features.

The third research question (Q3) investigated whether the extended UML use case diagram supported boundary changes. The answer to this question is yes. Whilst 4 of the changes to the use case diagram were possible with an existing UML diagram, the remaining 8 were made possible from the representation of system boundaries introduced into the notation. One alternative – to draw the UML diagram after each change – was not tractable in the workshop due to the number of changes made in a short time. In contrast, our notation extension – numbers and letters – was simple and quick to change during the workshop. We believe that this feature of a requirements notation is important for creative requirements processes, where system boundaries, allocation of work to actors and system features change rapidly.

There are 4 possible threats to the validity of the post-workshop idea rankings. The first is that the 3-month delay for ranking the idea might have caused stakeholders to forget the ideas, leading to incorrect ratings. However stakeholders continued to analyze the ideas after the workshop, ensuring familiarity with them. A second threat is that the 6 post-workshop respondents did not represent all VANTAGE stakeholders. However, over half of the workshop participants rated the workshop ideas. The third and fourth threats, however, cannot be dismissed so easily. We chose not to ask stakeholders to rate all of the workshop ideas to avoid overloading them with information about the use cases essential to their interpretation. Yet there remains a risk that these 24 ideas had characteristics, such as specific contexts of application, which might make them more useful in VANTAGE. Nonetheless, real-world constraints on the availability of the stakeholders led us to choose to capture incomplete data with some reliability rather than risk this data capture by seeking too much. The fourth threat was that we do not know whether the stakeholders knew many of the ideas generated during the workshop prior to it. Our decision to research the usefulness rather than the novelty of ideas meant that we did not provide a baseline for the workshop data. Another threat is that stakeholders ranked known ideas more familiar to them as more useful, leading to results that over-estimate the usefulness of the workshop. That said, one purpose of the day-1 brainstorming was to surface such known ideas, and only 34 ideas were documenting, thus providing some evidence that new ideas might have emerged during subsequent creative activities.

7 Discussion

This paper reports one workshop that delivered requirements and design ideas used in later requirements activities for a major airport operations system. In post-workshop analyses stakeholders rated just under one-third of the requirements and design features as definitely to be implemented, suggesting the usefulness of the workshop. Some creative techniques, such as constraint removal and brainstorming, generated more ideas that were ranked as to be implemented. The extended UML use case notation supported changes to system boundaries during early creative activities.

More requirements and design ideas generated in the workshop were useful to the airport and community beneficiaries than to the technology partners. One extrapolation is that the workshop facilitated the transfer of knowledge from these partners to airport and community stakeholders via the communication as well as the invention of ideas. Such communication appears to have been more successful during shared creativity activities such as constraint removal and storyboarding. If correct, this successful communication is in contrast to the technology presentations, during which the technology partners presented their solutions directly to other stakeholders but did not generate many ideas. This emerging role of knowledge communication will be considered in future refinements of the design of the workshops.

Results from the constraint removal suggest it is difficult to separate generation of useful and un-useful ideas. This result demonstrates an important lesson – that more focused creativity activities that seek to reduce “noise” – i.e. un-useful ideas – might not be as successful as those that do.

The results support previous findings [6, 7, 8] that storyboarding is an effective technique for combining and generating ideas for subsequent use case specification. Stakeholders trawled the boards to select ideas to include in each storyboard. As such this form of storyboarding appeared to be more flexible than originally anticipated.

Finally, extending the use case diagram notation to express and challenge system boundaries during exploratory creativity also worked as intended. We could have represented system boundaries as additional boundary rectangles. However, the simple-to-change tags were easier to change during the workshop, thus avoiding the viscosity of the notation [22]. New research is needed to develop new notations that support creative requirements activities, and we look forward to reporting these in the future.

Acknowledgements

The work was funded by the UK DTI-supported VANTAGE Phase-1 project.

References

1. Maiden, N.A.M., Manning, S., Robertson, S., Greenwood, J.: Integrating Creativity Workshops into Structured Requirements Processes. In: Proceedings DIS 2004, Cambridge Mass, pp. 113–122. ACM Press, New York (2004)
2. Maiden, N.A.M., Robertson, S.: Developing Use Cases and Scenarios in the Requirements Process. In: Proceedings 26th International Conference on Software Engineering, pp. 561–570. ACM Press, New York (2005)

3. Isaksen, G., Dorval, K.: Changing views of creative problem solving: Over 40 years of continuous improvement. *ICN Newsletter* 3(1) (1993)
4. Heitmeyer, C.: System Designers, Not Analysts Should Design. *IEEE Software* 22(1), 49–51 (2005)
5. Yu, E., Mylopoulos, J.M.: Understanding “Why” in Software Process Modeling, Analysis and Design. In: *Proceedings 16th International Conference on Software Engineering*, pp. 159–168. IEEE Computer Society Press, Los Alamitos (1994)
6. Maiden, N.A.M., Robertson, S.: Integrated Creativity into Requirements Processes: Experiences with an Air Traffic Management System. In: *Proceedings 13th International Conference on Requirements Engineering*, pp. 105–114. IEEE Computer Society, Los Alamitos (2005)
7. Maiden, N., Robertson, S., Gizikis, A.: Provoking Creativity: Imagine What Your Requirements Could be Like. *IEEE Software* 21(5), 68–75 (2004)
8. Maiden, N.A.M., Robertson, S., Ncube, C.: Can Requirements Be Creative? Experiences with an Enhanced Air Space Management System. Technical Report, Centre for HCI Design, City University, London, UK (2007)
9. Jones, S.V., Maiden, N.A.M.: RESCUE: An Integrated Method for Specifying Requirements for Complex Socio-Technical Systems. In: Mate, J.L., Silva, A. (eds.) *Requirements Engineering for Socio-Technical Systems*, pp. 245–265. Ideas Group (2005)
10. Floyd, C., Mehl, W.-M., Reisin, F.-M., Schmidt, G., Wolf, G.: Out of Scandinavia: Alternative Approaches to Software Design and System Development. *Human-Computer Interaction* 4(4), 253–350 (1989)
11. Osborn, A.F.: *Applied Imagination: Principles and Procedures of Creative Problem Solving*. Charles Scribener Sons, New York (1953)
12. Robertson, J.: Eureka! Why Analysts Should Invent Requirements. *IEEE Software* 19(4), 20–22 (2002)
13. Nguyen, L., Carroll, J.M., Swatman, P.A.: Supporting and Monitoring the Creativity of IS Personnel During the Requirements Engineering Process. In: *Proceedings Hawaii Int’l Conf. Systems Sciences*, vol. 33, IEEE Computer Society, Los Alamitos (2000)
14. Mich, L., Anesi, C., Berry, D.M.: Requirements Engineering and Creativity: An Innovative Approach Based on a Model of the Pragmatics of Communication. In: *Proceedings REFSQ 2004 Workshop*, Riga (2004)
15. Sternberg, R.J. (ed.): *Handbook of Creativity*. Cambridge University Press, Cambridge (1999)
16. Boden, M.A.: *The Creative Mind*, Abacus, London (1990)
17. Poincare, H.: *The Foundations of Science: Science and Hypothesis, The Value of Science, Science and Method*. University Press of America, Washington (1982)
18. Pennell, L., Maiden, N.A.M.: Creating Requirements – Techniques and Experiences in the Policing Domain. In: *Proceedings REFSQ 2003 Workshop*, Velden, Austria (June 2003)
19. Andrews, D.C.: JAD: A Crucial Dimension for Rapid Applications Development. *Journal of Systems Management*, 23–31 (1991)
20. Green, T.R.G.: Cognitive Dimensions of Notations. In: Sutcliffe, A., Macaulay, L. (eds.) *Proceedings HCI 1989, People & Computers V*, pp. 444–460. Cambridge University Press, Cambridge (1989)
21. <http://vega.soi.city.ac.uk/~cc559/REFSQ2008vFigure2.jpg>
22. <http://vega.soi.city.ac.uk/~cc559/REFSQ2008vFigure1.jpg>

A Stakeholder Model for Interorganizational Information Systems

Luciana C. Ballejos, Silvio M. Gonnet, and Jorge M. Montagna

CIDISI – INGAR

Instituto de Desarrollo y Diseño – Univ. Tecnológica Nacional – FRSF

Avellaneda 3657 – (3000) Santa Fe – Argentina

Tel/Fax: +54-342-4553439

{lballejos,sgonnet,mmontagna}@santafe-conicet.gov.ar

Abstract. Stakeholders constitute the principal source of requirements in the development of information systems. They therefore must be considered all over the process. In order to achieve success, they must be also modelled and then integrated with requirements, design and implementation models. Thus, a more complete perspective is added to traditional modelling. This work presents and describes a stakeholder model for interorganizational information systems, in order to incorporate a stakeholders-including approach to traditional modelling, focusing on interorganizational environments.

Keywords: stakeholder, interorganizational information systems.

1 Introduction

In requirements elicitation, the stakeholder concept is fundamental. Stakeholders are the primary requirements source for software projects [1]. They are defined as any group or individual that can affect or be affected by the attainment of organizational objectives or that must be involved in a project because is affected by its activities or results [2]. Each stakeholder has a unique view on the system. By means of their coordinated efforts the system is conceived, created and maintained.

Diverse changes of perspectives in organizational management and engineering areas are taking place by these days. Nowadays, various issues are changing the way of doing businesses. Organizations now tend to cooperate and create links with other organizations due to economic globalisation, changes in consumers needs and requirements, new market trends, ICTs dynamic development, etc., conforming what is known as Interorganizational Networks (IONs). Operations and interchanges between participant organizations are supported by a special type of information system: Interorganizational Information Systems (IOSs). They are the main tool to support and coordinate interorganizational (IO) processes and relations.

In the engineering area there is a change from design processes centered in the user towards more participative experiences. A new perspective has arisen from design FOR users towards design WITH users, where new ways of thinking and working are required. Participative design is not just a method or a set of methodologies but an

attitude towards people. It is based on the belief that all the people have something to offer to the design process. This approach is promoted by various authors. Sanders [3] affirms that persons want to express themselves and participate directly in design process development. Thus, the great challenge of creating tools and infrastructures to support and facilitate the design processes considering users experience is posed.

There exist diverse initiatives to disseminate this perspective, that requires the explicit representation of stakeholders in information system development models [4, 5]. This approach not only helps in the common understanding of the system design process, but also supports the coordinated effort required for its development, through the connection of the diverse activities which compose the process with the stakeholders capable to execute them. Also the satisfaction level is assured, since stakeholders needs are considered from early design stages. By representing stakeholders in systems models, diverse issues can be analyzed and addressed such as conflicts between stakeholders, rationale behind requirements, etc.

This approach is even more important in IO environments, where personal interactions are less frequent and more difficult due to the geographical dispersion that generally takes place between participants and where diverse cultures, interests, and points of view exist. Thus, considering the latent needs of (a) counting with tools and systems to support interorganizational linkages and (b) involving stakeholders in systems design processes, this article proposes a stakeholder model. It can be used not only in requirements modelling but also in other stages of the design process, thus obtaining stakeholders-including models and achieving a more complete vision of the process. This proposal helps in reducing the existing gap between what is the *problem domain*, formed by stakeholders and their needs, and the *solution domain*, which has its initial steps in the requirements and needs modelling associated to their main sources (stakeholders).

In order to present an orderly explanation of the model development, Section 2 characterizes IOSs and stakeholders for IO environments. Section 3 relates the stakeholder and the actor concepts and introduces the role concept, very important in process representation. Section 4 progressively develops and describes an integrated stakeholder model using the concepts explained in Sections 2 and 3, and introducing new properties. An example of the proposed model is included in Section 5. Finally, diverse conclusions of the work and new possible research lines are presented in Section 6.

2 Stakeholders and Interorganizational Information Systems

In the context of contemporary global economy, any design process, and more strongly IOSs design processes, implies multiple teams and stakeholders collaborating for attaining a common goal. Being able to capture efficiently and clearly their needs is increasingly more important and complex.

As opposed to traditional environments, in IO environments stakeholders are more numerous and their interests vary considerably. They are defined as “any individual, group or organization which can affect or be affected (positively or negatively) by the system under study and which has direct or indirect influence on its requirements” [6, 7, 8]. Fig. 1 represents the concepts under this definition, where a *stakeholder* can

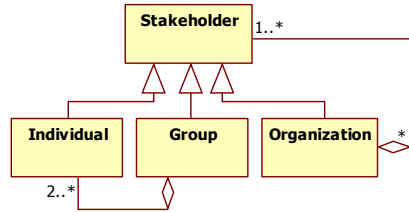


Fig. 1. Stakeholder Concept

be an *individual*, a *group* or an *organization*, where an *organization* is composed by one or more *stakeholder individuals*, *groups* or *organizations*. In general, a *group* is an aggregation of -at least- two *individuals*.

3 Stakeholders, Actors, and Roles

Pfahl [9] considers the actor concept as essential in order to represent processes models besides activities, artefacts (which are used and produced by activities), tools (which are used by activities), and roles (which carry out activities). This concept defines responsibilities between agents and activities of particular processes.

Nevertheless, a subtle difference between the terms *actor* and *stakeholder* exists. Stakeholders are those which have some interest in the process and will be affected positively or negatively by the results to be obtained. Thus, the set of stakeholders of a particular process is more numerous and, at the same time, **includes** the set of actors of that process.

In general, process modelling is limited to represent only individuals who will directly execute activities. Also, in any domain, the execution of activities by actors is restricted to the roles they may play in particular moments. While an actor represents a specific entity (individual, group, or computational program), a role represents a position which might be played by diverse actors. Also an actor might be associated to more than one role and also a role can be played by more than one actor. A role implies the possibility or capacity to execute a set of activities. Van Welie and van der Veer [10] define it as a collection of tasks performed by one or more agents. The tasks might be hierarchically decomposed. Actor's roles analyze and consider their responsibilities on the project and their relation with the artefact or final product to be obtained as result. In the case of information systems, for example, roles arise from the analysis of the possible interaction types which can exist between a particular stakeholder and the future system to be developed.

Thus, the role concept avoids personifying the relation between actors (stakeholders) and activities and is a very useful concept to model properties and behaviours of entities which evolve over time in processes models [11]. Methodologies can be easily described and planned through profiles that can be assumed by the participating entities.

Kueng et al. [12] describe two strengths of the role concept: (1) during modelling stage, abilities, functionalities, competencies and responsibilities must not be discussed,

and (2) during operative stage, when the model is used, entities with the same role are potentially interchangeable.

Role concept can be transferred to collaborative design area, which involves stakeholders with different intentions, formations and knowledges, and where activities are influenced not only by technical decisions, but also by social interactions [13]. Roles and stakeholders analysis and modelling introduce elements of Social Sciences in the representations, shaping a complementary dimension to traditional ones for design processes [14].

Stakeholder analysis provides a baseline for effective requirements engineering and subsequent system design, as well as for eliciting requirements for all key stakeholders. Macaulay et al. [15], Kirby [16] propose approaches to determine the major categories of stakeholders for an information system. Similarly, Robertson [17] and Alexander and Robertson [18] present a well-explained model describing diverse categories of stakeholders using “the onion model” and locating each category in one of the “onion levels” (rings). Nevertheless, in IO environments one more step is needed towards the consideration of certain issues related to those contexts (e.g. the interorganizational dimension is added, from which stakeholders can also exist). Furthermore, some more work towards stakeholders concrete selection must be done. Taking into account this problem, Ballejos and Montagna [19] have proposed a method for stakeholders identification for IOSs.

When identifying the concrete stakeholders for any software project, diverse attributes and properties related with the project are also determined: dimension to which they pertain, roles, and interest and influence degrees. The assignment of roles to stakeholders describes their relation to IOS design process. It also allows an easier stakeholder management, by grouping them through roles. In this way, stakeholders sharing the same profile in relation to IOS can be managed altogether.

Zhang and Chen [14] pose that a clear identification of stakeholders roles and their participation degree in the diverse design stages are important steps towards the success of distributed collaborative design. From this affirmation, the first concepts related to the model development must be addressed in order to represent stakeholders (Fig. 2). The *role* (operator, regulator, responsible, beneficiary, etc.) represents the relation between the stakeholder and the design process activities.

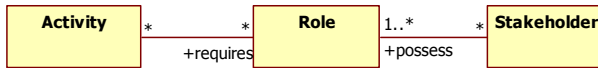


Fig. 2. Stakeholders and Roles

Fig. 2 takes into account that every *stakeholder* has at least one *role* (1..*) and that a *role* can be played by diverse *stakeholders* (zero or more, as indicated by *). The association-end *possess* indicates the stakeholder position in relation to the process or project under analysis.

On the other hand, the execution of activities by stakeholders is performed through the assignment and utilization of roles. Thus, the model must also include the necessary concepts in order to represent that every *activity requires* particular *roles* to be executed and that every *role* is required for the execution of some *activity* (Fig. 2).

Between the approaches which incorporate the role concept, activities are executed only when certain abilities are possessed. In this context, Gonnet et al. [20] use the *skill* concept, while Harzallah and Vernadat [21] refer to *competency* when making reference to the attribute needed to meet a mission or execute an activity in their formal models. However, this can be generalized considering the actor concept by Ellis and Wainer [22]: “an actor is a person, computational program or entity which must play roles to **execute, be responsible for, or be associated in some manner** with activities and procedures”. Analyzing this concept and comparing it with the IO stakeholder presented in Section 2, it can be deduced that this actor concept for Ellis and Wainer corresponds to stakeholder one. So, it can be used in order to represent participative environments, where other criteria also exist when relating stakeholders with activities, and not only the ability. For example, functions performed, hierarchical level, geographical location, etc. are attributes independent from the ability or specific knowledge of the individuals. In the model, these specific properties will be materialized through the role concept, such as showed by Fig. 2.

However, as it was previously stated, a stakeholder executes a particular activity playing a certain role. Thus, to count with information related to this, an association-class is needed in order to integrate the information regarding the execution of a particular activity. In Fig. 3, *execution* association-class contains the role played by the stakeholder when executing an activity.

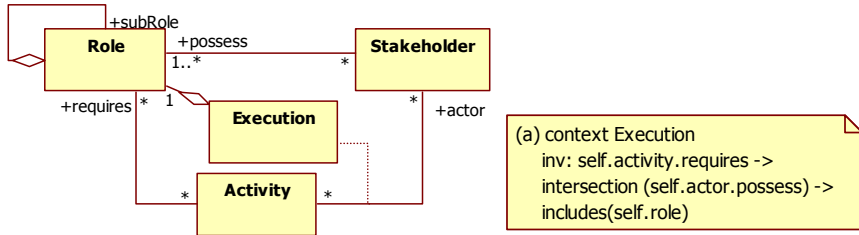


Fig. 3. Stakeholders, Activities, and Roles

Fig. 3 integrates the concepts related with the position of a determined stakeholder (*possess* association-end), activities management (*requires* association-end) and activities execution by a stakeholder playing a particular role (*execution* class). *Execution* association-class indicates, in the moment to be instantiated the model, the *role* possessed in that instant by the *stakeholder* executing an *activity*.

A *subrole* relationship indicates that a *role* includes other *roles*, including also their relations with *activities*. An example is the beneficiary role, whose subroles are: functional, financial, political, and sponsor. Thus, all activities required to be executed by a beneficiary role, may be also executed by all its subroles.

An OCL restriction is included in the model in order to indicate that certain role is possessed by an actor and is also required for an activity to be executed. In other words, stakeholders only can execute an activity when they possess the roles required by that activity.

In conclusion, far from being redundant, the *Execution* class included in Fig. 3 guarantees the independence of concepts: *stakeholder possess role*, *activity requires*

role and *stakeholder* with certain *role* executes *activity*. Their integration gives responses to questions involving all of them, such as: which *stakeholders* can execute certain *activity*? (knowing that the activity can be executed only by certain roles), which *activities* can certain *stakeholder* execute? (having the *stakeholder* certain *roles* associated), which *roles* has a certain *stakeholder* who can execute a certain *activity*? (knowing that an *activity* can be executed by certain *roles*), or under which *role* had a *stakeholder* executed certain *activity*?

4 Stakeholder Model

Stakeholders are the main concept to be considered and represented in order to create complete design models. This idea is more significant in IO projects, where shared objectives are more diffuse and requirements management is more complex.

Once stakeholders to be involved in requirements elicitation are selected, besides counting with basic information about them, descriptive attributes such as roles, interest, and influence are also known.

Interest derives from the relation between stakeholders needs and project goals. Fig. 4 models the concepts associated to a stakeholder interest and determines that the existence of certain *interest* promotes zero or more project goals (*projectGoal*).



Fig. 4. Interest Model

Information regarding stakeholders interest will be very useful in future modelling stages, when *requirements* and their properties will be associated to project goals. Then, diverse influence analyses could be executed over stakeholders interests when managing requirements and their properties (Fig. 4 only shows *requirements* concept to give a general understanding, avoiding the concepts related to it).

On the other hand, **influence** indicates the stakeholder relative power on the project and the decisions which must be taken about it. In general, when stakeholders are analyzed, authors generally describe two levels in which interest and influence can take place: high and low [23, 24]. So, an initial estimation of the priority associated to requirements is attained. Other authors specify a scale to be used, in order to provide more utility to particular analysis. For example, Bourne and Walker [25] use five values in the range between “very high” and “very low” in order to obtain an intensity index of stakeholders interest.

In some sense, stakeholder roles represent a relation between stakeholders and the project. They can be associated with certain influence or decision power over the project, independently from the particular stakeholder who might play the role. Table 1 presents examples of common stakeholders roles for information systems development projects. From it, the influence degree or each role over the project might be deduced. For example, *responsible*, *decision-maker*, and *regulator* are roles with greater influence than the one associated to *operator*, *consultant* or *functional*

beneficiary. Thus, in the determination of stakeholders influences, their associated roles must be analyzed.

The analysis of each role defined in Table 1 in relation to its possible influence on the project brings out a new property, *roleInfluence*. Through *roleInfluence* such relation is dimensioned and calculated through qualitative (e.g.: high, medium, low) or quantitative (e.g.: 1, 2, 3; or 1, 5, 10) values. Table 2 shows possible influence degrees associated to roles, where High \rightarrow 3, Medium \rightarrow 2, and Low \rightarrow 1.

Table 1. Stakeholders Roles

Beneficiary: Those that benefit from system implementation. They can be: functional, financial, political sponsors.
Negative: Those that undergo some kind of damage or are adversely impacted by system development.
Responsible: They are in charge of the system in all its lifecycle phases. This role includes people working with budgets and schedules (for example, project manager, those responsible for selecting suppliers, etc.)
Decision-Maker: Those that control the process and make decisions to reach agreements.
Regulators: They generate guidelines and outlines that will affect the system development or operation.
Operators: They interact with the system and use its results (information, products, etc.).
Experts: They widely know the implementation domain and can collaborate in requirements elicitation.
Consultants: Include any role dealing with providing support for any aspect of the system development.
Developer: requirements engineer, analyst, programmer, tester, security engineer, project manager, etc.

Table 2. Stakeholders Roles and associated Influences

Role		RoleInfluence	
Beneficiary	Functional	Low	1
	Financial	Medium	2
	Political	Medium	2
	Sponsor	High	3
Negative		Medium	2
Responsible		Medium	2
Decision-Maker		High	3
Regulator		High	3
Operator		Low	1
Expert		Medium	2
Consultant		Low	1
Developer		Medium	2

Bourne and Walker [25] affirm that *power* sources determine the stakeholder influence. Yukl [26] defines three possible stakeholder power sources: *positional power*, derived from authority (e.g., organizational), *personal power*, derived from influence on human relationships or specific features such as experience, charisma, loyalty/friendship, etc., and *political power*, derived from control positions over decision processes in relation to the particular project. Thus, once the stakeholders are selected, project manager must determine the power sources associated to each one. Also, the different power sources might be associated to diverse qualitative and quantitative values previously selected by project manager. Fig. 5 outlines the representation of the concepts. The **ppow** attribute for Power types is a value

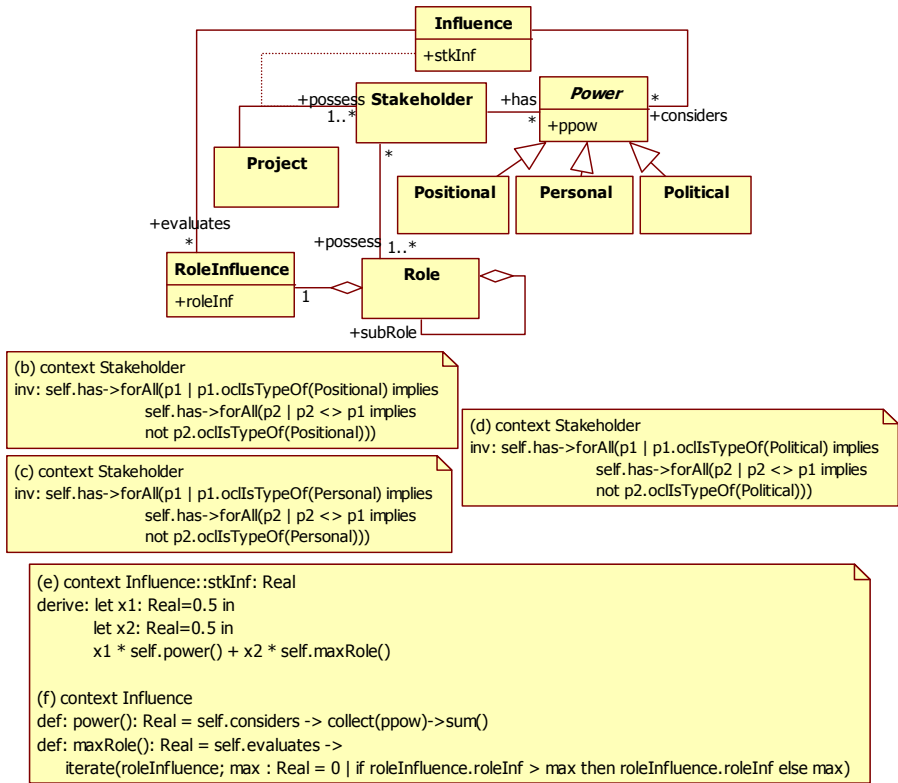


Fig. 5. Power and Influence Model

indicating the importance assigned to each power source by the project manager. OCL restrictions (b), (c), and (d) indicate that any stakeholder can be associated with a unique occurrence of each power source type.

According to the model, the *influence* of a particular *stakeholder* on the *project* is obtained from a weighting function applied over stakeholder *power* sources and influence values of each assigned role (*roleInfluence*) (see (e) OCL restriction in Fig. 5). The weight assigned to each attribute depends on its importance and on the criteria adopted by the project manager in order to consider both or not to assess the influence. In this way, influence values will be obtained from (1), where **Power** is the addition of *ppow* values of each stakeholder associated power type and **Max(roleInfluence)** is the maximum value of all *roleInfluence* values of *roles* associated to the *stakeholder* (see (f) OCL restriction in Fig. 5). Another mathematical function might be used instead of maximum, for example, average. Thus, a concrete value for representing a stakeholder influence is obtained considering, not only the value of the roles assigned (*roleInfluence*) –which is independent from the individual, group or organization selected as stakeholder–, but also a value given by the analysis of the specific individual, group or organization selected as stakeholder (*power*), independently from their assigned roles.

$$\text{Influence} = x_1 * \text{Power} + x_2 * \text{Max}(\text{roleInfluence}) . \quad (1)$$

Where: $0 \leq x_1 \leq 1$, $0 \leq x_2 \leq 1$, and $x_1 + x_2 = 1$.

On the other hand, **dimension** is a significant concept for IO environments. In traditional environments, stakeholders dimension is determined by considering if stakeholders belong to the organization under study or not, classifying them as internal or external. However, in IO environments a new dimension must be generated, the interorganizational one. Thus, stakeholders may exist in three possible dimensions: internal or organizational (whose stakeholders represent the interests of a particular organization), interorganizational (stakeholders pursue ION goals and may pertain or not to an ION participant organization), and external (representing interests from ION external entities).

Fig. 6 considers that a stakeholder may have one or more dimensions. Particular stakeholders might be selected to represent their organization (*organizational dimension*), and at the same time, the ION (*interorganizational dimension*) where the organization takes part, considering the interorganizational interests. Also the same stakeholder may represent more than one ION organization, thus having also their dimensions associated.

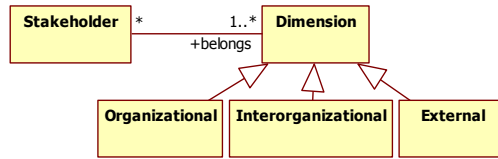


Fig. 6. Dimension Model

This information is very important to determine requirements contexts, when they are modelled in relation to stakeholders. Thus, organizational, interorganizational or external requirements might be discovered when analyzing the dimensions of their source stakeholders.

Thus, considering on the one hand stakeholder concept and properties previously described, and, on the other hand, the execution of activities associated with roles, Fig. 7 proposes a model for stakeholders.

Interest and *influence* concepts are critical due to their dynamism. They might also be affected by the variation of *roles* for a specific *stakeholder* during the project. They may change over time due to diverse factors: political, cultural, etc. Thus, they must be analyzed, for example, when prioritizing and managing requirements and when conflicts between stakeholders requirements exist.

The inclusion of stakeholder *dimension* enables organizational, interorganizational and external stakeholders modelling, the management of requirements is improved and they may be also grouped through these dimensions.

Following Gonnet et al. [20] idea for design process modelling, each *stakeholder* may have *interests* which express intentions and particular desires. Also, those interests promote in some manner project goals which are directly related to requirements. Also stakeholders are directly related to them. Thus, the subsequent requirements management and analysis will give information regarding which

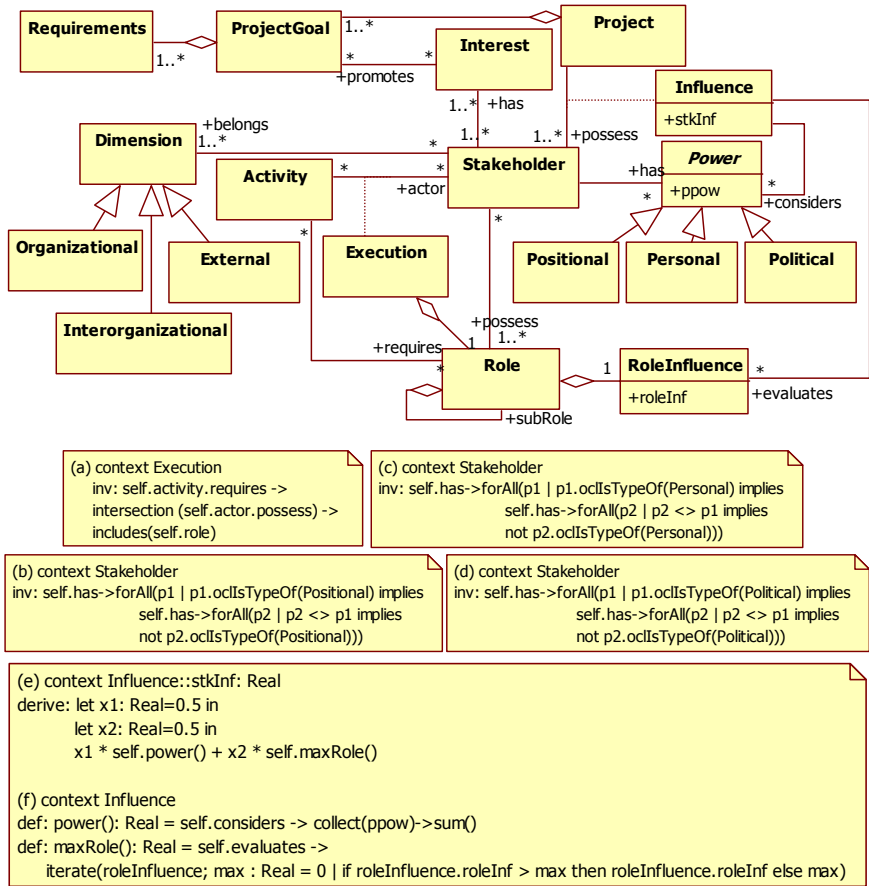


Fig. 7. Stakeholders Representation Model

requirements helps to attain project goals and in which manner. This information will help in deducing which stakeholders interests are being considered when satisfying some requirement.

5 Example

In the Public Health Area of an Argentinian province, in order to satisfy health primary needs and manage medicines and drugs distribution to health centers, an ION was created, which is shown in Fig. 8.

- Medicines Producer Laboratory (MPL) elaborates generic medicines at a low cost, to be provided to the population in health centres. Its unique customer is the Central Pharmacy.
- Central Pharmacy (CP) depends on the Provincial Health Department. Its goals are to plan, coordinate and control the supply of medicines and other elements required

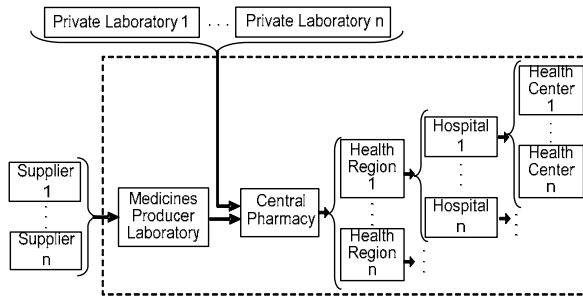


Fig. 8. ION for Medicines Production and Distribution

by health centers. The MPL is one of its principal suppliers. Private laboratories also provide medicines to the CP.

- Regional Health Areas. The province is divided in 9 health regions, each one responsible for medicines distribution in hospitals and centres depending on them.
- Hospitals and Health Centers.
- At external level, drugs suppliers, patients, other government areas.

The need of information integration has accelerated an IOS development and implementation for managing the wide set of interactions that cover generation, movement and access to medicines and information all around the state. The main goal is the transformation of the current model of separated organizational systems into a globalizing model over the ION described.

Diverse stakeholders were identified through the application of the approach proposed by Ballejos and Montagna [19]. Some of them and their attributes are described in Table 3, where the last column describes the type of power each stakeholder has.

As is shown in Table 3, diverse stakeholders can be selected from the case study. They are associated with diverse roles and predominant power types.

The next step is to average the influences of each stakeholder roles. Table 4 presents the results for each stakeholder. Also some value or importance degree must be assigned to different power types by project manager. For example, in a project where positional power is more important than personal power and this last one is more important than political one, values like 3, 2 and 1 can be assigned respectively. All of them can be assigned the same value if they are equally important.

Then, the project manager must decide on specific weights for both **power** types and the influence arisen from stakeholders roles respectively, in order to determine each stakeholder influence in the project. Thus, for example, if the same importance is given to both, the calculus must use the function: **Influence = 0.5*Power + 0.5*Max(roleInfluence)**, and a table like the one presented in Table 5 is the result for the example, where Influence values range from 0 -indicating the lowest degree of influence to 3 -indicating the highest degree of influence-.

Diverse object-models can be derived with the information of the example. An instantiation from the model in Fig. 7 is presented in Fig. 9 for the stakeholder “Central Pharmacy Director”.

Table 3. Stakeholders attributes for the example

Stakeholder	Dimension	Role/s	Power
Central Pharmacy Director	Organizational Interorganizacional	• Political Benefic. • Decision-Maker • Responsible	Positional Political
Pharmacy Department Employees from each Hospital	Organizational	• Functional Benefic. • Operator	--
Central Pharmacy Purchase Manager	Organizational	• Operator • Functional Benefic.	Personal
Central Pharmacy Operative Staff	Organizational	• Operator • Functional Benefic.	Personal
Administrative Employees from each Health Center	Organizational	• Operator • Negative	--
Provincial Health Department	Interorganizacional	• Political Benefic. • Financial Benefic. • Regulator	Political
Health Region Coordinator	Interorganizacional	• Political Benefic. • Responsible	Positional Personal
Patients	External	• Functional Benefic.	--

Table 4. Stakeholders and their roles influences for the example

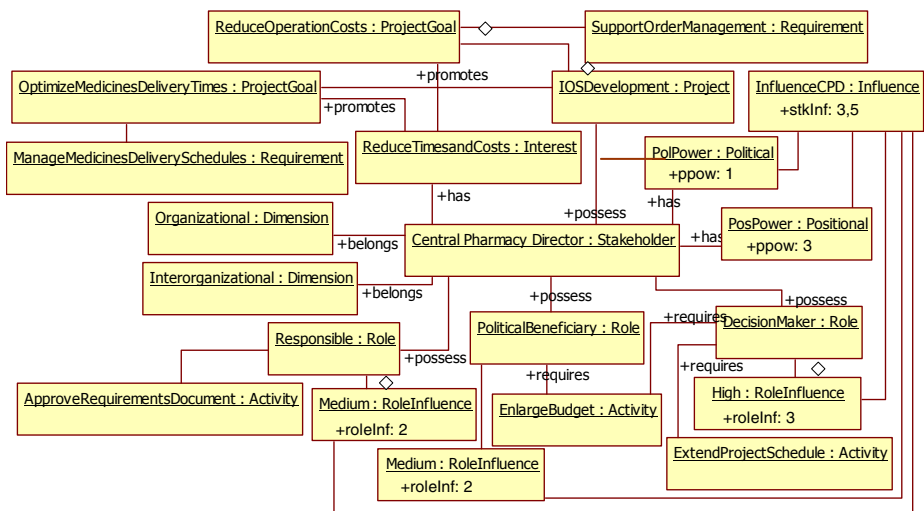
Stakeholder	Roles	Role Influence
Central Pharmacy Director	• Political Benefic.	2
	• Decision-Maker	3
	• Responsible	2
Pharmacy Department Employees from each Hospital	• Func. Benefic.	1
	• Operator	1
Central Pharmacy Purchase Manager	• Operator	1
	• Funct. Benefic.	1
Central Pharmacy Operative Staff	• Operator	1
	• Funct. Benefic.	1
Administrative Employees from each Health Center	• Operator	1
	• Negative	2
Provincial Health Department	• Political Benefic.	2
	• Financ. Benefic.	2
	• Regulator	3
Health Region Coordinator	• Political Benefic.	2
	• Responsible	2
Patients	• Funct. Benefic.	1

In Fig. 9, “Central Pharmacy Director” stakeholder belonging to “organizational” and “interorganizational” dimensions is associated with “Reduce Time and Costs” interest. Also the corresponding influence is calculated through the previously described power and role influences values of his associated roles. In this case, the stakeholder has the highest influence value.

“Reduce Time and Costs” interest promotes “Reduce Operation Costs” and “Optimize Medicines Distribution Times” project goals, which are directly associated with “Support Orders Management” and “Manage Medicines Delivery Schedules” requirements for the IOS.

Table 5. Stakeholders Influence for the example

Stakeholder	Max (RoleInfluence)	Power	Influence
Central Pharmacy Director	3	$\frac{(3+1)}{4}$	3,5
Pharmacy Department Employees from each Hospital	1	0	0,5
Central Pharmacy Purchase Manager	1	3	2
Central Pharmacy Operative Staff	1	3	2
Administrative Employees from each Health Center	2	0	1
Provincial Health Department	3	3	3
Health Region Coordinator	2	$\frac{(3+2)}{5}$	4,5
Patients	1	0	0,5

**Fig. 9.** Object model for “Central Pharmacy Director” stakeholder

In this way, the model describes roles, power and influence assigned to the selected stakeholder. It also associates influences for each played role. In stakeholder management, diverse activities might also be associated to certain roles, in order to have control over activities execution in design processes. Also, possible conflicts in stakeholders interests may be assessed with the instantiation of the complete model with the information of all existing stakeholders. Also, requirements management effects over stakeholders might be analyzed. So, the proposed model allows not only a better understanding of the situation underlying requirements and their source stakeholders in interorganizational environments, but also the execution of diverse evaluations, useful in managing stakeholders and requirements throughout the software development process.

6 Conclusions and Future Works

This article has merged two important areas in information systems engineering: on the one hand, the development of information systems with the latent need of involving stakeholders in the process, on the other hand, the current and constant emergence of interorganizational relationships needing to be technologically supported. This is a first step towards reducing the existing gap between stakeholders needs (problem domain) and system requirements (solution domain) by proposing a model for representing stakeholders and their needs, in order to include them in the requirements model. The model also considers diverse stakeholders properties which have incidence in their management and in requirements analysis also. It allows a complete understanding of the environment through the modelling of their principal stakeholders interests. Thus, not only requirements could be clearly managed, but also conflicts between stakeholders can be detected and handled.

Design models arise from closed requirements specifications. However, there are no models for analyzing the rationale after the design, where diverse stakeholders decisions and activities derive in the requirements specification, and main source of information for design stage. Thus, the proposed model integrated with the requirements model will constitute the basis for future research towards the analysis of the rationale behind the requirements management stage in interorganizational information systems development. It will also enable influences analysis, thus obtaining a more reality-adjusted model of the underlying knowledge.

Acknowledgements

The authors want to thank the financial support from CONICET, ANPCyT, UTN and Universia-Banco Río.

References

1. Bittner, K., Spence, I.: Establishing the Vision for Use Case Modeling, Use Case Modeling. Addison Wesley Professional, Reading (2003)
2. Pouloudi, A., Gandeche, R., Papazafeiropoulou, A., Atkinson, C.: How Stakeholder Analysis can Assist Actor-Network Theory to Understand Actors. In: A Case Study of the Integrated Care Record Service (ICRS), UK National Health Service Eltrun Working Paper Series, WP 2004-002 (2004)
3. Sanders, E.B.N.: From User-Centered to Participatory Design Approaches. In: Frascara, J. (ed.) Design and the Social Sciences, Taylor & Francis Books Limited, Abington (2002)
4. Mostashari, A., Sussman, J.: Engaging Stakeholders in Engineering Systems Representation and Modeling. In: Engineering Systems External Symposium, Massachusetts Institute of Technology (2004)
5. Carpenter, S.: Choosing Appropriate Consensus Building Techniques and Strategies. In: Susskind, L., McKernan, S., Thomas-Larmer, J. (eds.) The Consensus Building Handbook: A Comprehensive Guide to Reaching Agreement, pp. 61–97. Sage Publications, Thousand Oaks (1999)
6. Kotonya, G., Sommerville, I.: Requirements Engineering: Processes and Techniques. J. Wiley & Sons(eds.) (2003)

7. Pouloudi, A.: Aspects of the Stakeholder Concept and their Implications for Information Systems Development. In: 32th Annual Hawaii International Conference on System Sciences (1999)
8. Sharp, H., Finkelstein, A., Galal, G.: Stakeholder Identification in the Requirements Engineering Process. In: Database & Expert System Applications, DEXA Workshop 1999, pp. 387–391. IEEE Press, New York (1999)
9. Pfahl, D.: An Integrated Approach to Simulation-Based Learning in Support of Strategic and Project Management in Software Organisations. PhD Thesis. Fraunhofer-Institut für Experimentelles Software Engineering (Fraunhofer IESE), Kaiserslautern (2001)
10. van Welie, M., van der Veer, G.C., Eliëns, A.: An Ontology for Task World Models. In: Markopoulos, P., Johnson, P. (eds.) DSV-IS 1998. LNCS, pp. 57–71. Springer, Vienna (1998)
11. Rabentós, R., Cabot, J.: Conceptual Modelling Patterns for Roles. In: Spaccapietra, S., Atzeni, P., Chu, W.W., Catarci, T., Sycara, K.P. (eds.) Journal on Data Semantics V. LNCS, vol. 3870, pp. 158–184. Springer, Heidelberg (2006)
12. Kueng, P., Kawalek, P., Bichler, P.: How to compose an Object-Oriented Business Process Model? In: IFIP TC8, WG8.1/8.2 working conference on Method engineering: principles of method construction and tool support, UK, pp. 94–110. Chapman & Hall, Ltd., London (1996)
13. Lu, S.C.Y., Cai, J., Burkett, W., Udwadi, F.: A Methodology for Collaborative Design Process and Conflict Analysis. CIRP Annals-Manufacturing Technology 49(1), 69–73 (2000)
14. Zhang, H., Chen, D.: Developing a Multidisciplinary Approach for Concurrent Engineering and Collaborative Design I. In: Shen, W., Lin, Z., Barthès, J.-P.A., Li, T. (eds.) CSCWD 2004. LNCS, vol. 3168, pp. 230–241. Springer, Heidelberg (2005)
15. Macaulay, L., Fowler, C., Kirby, M., Hutt, A.: USTM: A New Approach to Requirements Specification. Interacting with Computers 2(1), 92–118 (1990)
16. Kirby, M.: Custom Manual, Technical Report DPO/STD/1.0, HCI Research Centre, University of Huddersfield (1991)
17. Robertson, S.: Project Sociology: Identifying and involving the stakeholders. The Atlantic Systems Guild Ltd (2000)
18. Alexander, I., Robertson, S.: Understanding Project Sociology by Modeling Stakeholders. IEEE Software 21(1), 23–27 (2004)
19. Ballejos, L., Montagna, J.M.: Stakeholders Selection for Interorganizational Systems: A Systematic Approach. In: Avison, D., Elliot, S., Krogstie, J., Heje, J.P. (eds.) The Past and Future of Information Systems: 1976–2006 and Beyond, vol. 214, pp. 39–50. Springer, Boston (2006)
20. Gonnet, S., Henning, G., Leone, H.: A model for capturing and representing the engineering design process. Expert Systems with Applications 33(4), 881–902 (2007)
21. Harzallah, M., Vernadat, F.: IT-based competency modeling and management: from theory to practice in enterprise engineering and operations. Computers in Industry 48(2), 157–179 (2002)
22. Ellis, C.A., Wainer, J.: A conceptual model of groupware. In: 1994 ACM conference on Computer supported cooperative work, pp. 79–88. ACM, New York (1994)
23. Applegate, L.M.: Stakeholder Analysis Tool. Harvard Business Online (2003)
24. Smith, L.W.: Project Clarity Through Stakeholder Analysis. The Journal of Defense Software Engineering (2000), <http://www.stsc.hill.af.mil/crosstalk/2000/12/smith.html>
25. Bourne, L., Walker, D.H.T.: Visualising and Mapping Stakeholder Influence. Management Decision 43(5), 649–660 (2005)
26. Yukl, G.: Leadership in Organizations. Prentice Hall International Inc., USA (2001)

Search Based Requirements Optimisation: Existing Work and Challenges

Yuanyuan Zhang¹, Anthony Finkelstein², and Mark Harman¹

¹ King's College London
Strand, London
WC2R 2LS, UK

² University College London
Malet Place, London
WC1E 6BT, UK

Abstract. In this position paper, we argue that search based software engineering techniques can be applied to the optimisation problem during the requirements analysis phase. Search based techniques offer significant advantages; they can be used to seek robust, scalable solutions, to perform sensitivity analysis, to yield insight and provide feedback explaining choices to the decision maker. This position paper overviews existing achievements and sets out future challenges.

1 Introduction

Once an initial set of requirements has been gathered by requirements elicitation, there is a business-level analysis problem: choices have to be made to identify optimal choices and trade-offs for decision makers. For example, one important goal is to select near optimal subsets from all possible requirements to satisfy the demands of customers, while at the same time making sure that there are sufficient resources to undertake the selected tasks.

To illustrate, Figure 1 demonstrates a possible spread of equally optimal requirements optimisation results. Two competing objectives are considered: cost to the provider and estimated satisfaction rating achieved by a solution. Each circle on the represents an equally optimal solution. That is, each circle denotes a solution for which no better solution (subset of requirements) can be found that offers better customer satisfaction without increasing cost. The set of possible solutions form what is known as a Pareto front. Pareto fronts show a solution space of candidate solutions, from which the decision maker can select. As will be seen later, Pareto fronts also yield insights into the structure of the problem.

This requirement selection problem is one example of the way in which requirements decisions can be formulated as optimisation problems. Other examples include ordering requirements to achieve earliest satisfaction, balancing each customer's needs against the others and balancing tensions between system and user requirements.

Such problems are inherently complex optimisation problems that seek to balance many competing and conflicting concerns, so it would be natural to

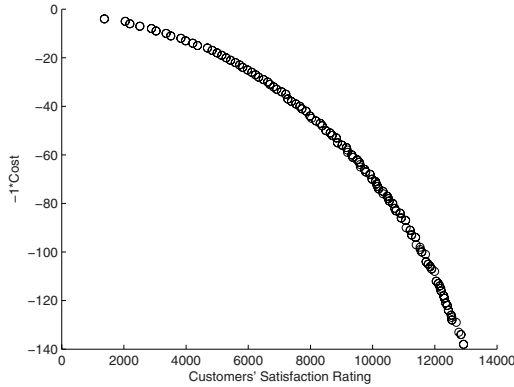


Fig. 1. Fictitious Data: 15 customers; 40 requirements. Adapted from Zhang et al. [13]. Each circle represents an equally optimal candidate solution that balances the objective of minimising supplier cost against the objective of maximising customer satisfaction. See Figure 2 for a comparison to real world requirements data from Motorola.

seek algorithms for decision support. Sadly is often infeasible to apply precise analytic algorithms, because the problems are typically NP hard. To overcome this difficulty, Search Based Software Engineering (SBSE) uses metaheuristic optimisation algorithms that explore and solve complex, multi-objective, highly constrained problems in Software Engineering [5]. This paper argues that Requirements Optimisation can be viewed as an application area for SBSE.

2 Background: Requirements Optimisation

Previous work on Requirements Optimisation has shown that metaheuristic optimisation techniques can be used to search for a balance between costs and benefits associated with sets of requirements. This has come to be known as the Next Release Problem (NRP) [2]. In the NRP, as formulated by Bagnall et al., the goal is to find the ideal set of requirements that balance customer requests within resource constraints.

In this formulation the problem is a constrained single objective optimisation problem. Bagnall et al. applied a variety of techniques to a set of synthetic data to demonstrate the feasibility of SBSE for this problem. Greer and Ruhe also studied the NRP [4], proposing an iterative Genetic Algorithm and presenting results for real world requirements problems. Their approach balances the resources required for all releases; assessing and optimizing the extent to which the ordering conflicts with stakeholder priorities.

More recently, there has been work on multi-objective formulations of the NRP [11,13]. In the multi-objective formulation, each of the objectives to be optimised is treated as a separate goal in its own right; the multiple objectives are not combined into a single (weighted) objective function. This allows the

optimisation algorithm to explore the Pareto front of non-dominated solutions. Each of these non-dominated solutions denotes a possible assignment of requirements that maximizes all objectives without compromising on the maximization of the others.

Zhang et al. [13] considered the two objectives of cost to the provider and estimated satisfaction rating for the customer, while Ruhe and Omolade [11] considered the two objectives that balance the tension between user-level and system-level requirements.

3 Advantages of the Search Based Approach

This section describes some of the ways in which SBSE techniques have proved to be effective in Requirements Optimisation and closely related problems.

Robustness. Software engineering problems are typically ‘messy’ problems in which the available information is often incomplete, sometimes vague and almost always subject to a high degree of change (including unforeseen change). Requirements change frequently, and small changes in the initial stages often lead to large changes to the solutions, affecting the solution complexity and making the results of these initial stages potentially fragile.

An important contribution of SBSE techniques is the way in which they can take changing factors and constraints into account in solution construction. They can, for example, provide near optimal solutions in the search space which remain near-optimal under change, rather than seeking optimal but fragile solutions [7]. This better matches the reality of most software projects, in which robustness under change is often as valuable as any other objective.

Sensitivity Analysis. In SBSE, human effort is partly replaced by metaheuristic search. Nevertheless, the numerical data upon which the automated part of the process depends come from expert domain knowledge. In the case of requirements engineering, the decision maker is forced to rely on estimates of these crucial inputs to the requirements optimisation process. Sensitivity analysis helps the developer build confidence in the model by studying the uncertainties that are often associated with parameters in models. It aims to identify how ‘sensitive’ the model is to change. This allows the decision maker to pay additional attention to estimates for which the model is particularly sensitive.

Insight. Requirements Optimisation problem instances have structure. That is, the data have implicit characteristics that the decision maker needs to expose in order to inform decision making. For any non-trivial problem, however, the number of requirements, customers, their interactions and dependencies make these implicit properties far from obvious. No human could be expected to simply look and see all the implications and important features of a problem instance. For example, the search may make it easier to see that satisfaction of one customer tends to lead to dissatisfaction of another or that requirement R_i is always in generated solutions in which R_j is present.

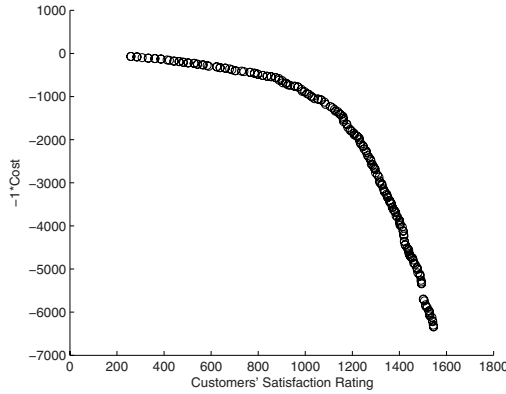


Fig. 2. Motorola mobile device requirements: 4 customers; 35 requirements. The optimisation produces a Pareto front of candidate solutions which reveals an important elbow point at cost 1,200.

In order to show how SBSE can yield insight in Requirements Optimisation, we now apply the cost-satisfaction formulation of Zhang et al. [13] to real requirement data from Motorola. The results are shown in Figure 2. These results have been anonymised to prevent disclosure of sensitive information.

Compare the real world results of Figure 2 with the smooth Pareto front in Figure 1. There is an ‘elbow point’ in Figure 2’s Pareto front which reveals a potential for optimisation: The customers’ satisfaction can be increased from 200 to approximately 1,200 at cost 2,000. This would be more attractive than the increase in satisfaction from 1,200 to 1,500, which would cost almost 3 times as much. The search has revealed a very attractive elbow point at cost 1,200. This kind of insight is very hard to achieve without automated optimisation, like that provided by such a search based approach.

Requirements Prioritisation. In the NRP, the decision maker not only selects the optimal or near optimal subset of requirements, but also the priority ordering of requirements. This method offers the potential for risk reduction. That is, circumstances vary and resources may become insufficient to fulfill all requirements. Prioritisation ensures that the most important requirements will be released first so that the maximum attainable benefits of the new system are gained earliest.

Fairness in Requirements Assignment. In the Requirements Optimisation process, it may be helpful to explore the extent to which the obtainable solutions can be said to be fair. Of course, fairness can come in different forms: should we spend the same amount on each customer or give each the same number or value of fulfilled requirements? Each notion of fairness can also be treated as a separate objective, for which a Pareto optimal search seeks non-dominated solutions [10]. In this way it becomes possible to automatically investigate the extent to which several notions of fairness can simultaneously be satisfied.

4 Challenges

This section describes some of the challenges for Search Based Requirements Optimisation.

Scalability. In Requirements Optimisation, problems arise not merely because of the number of requirements, customers and other participating factors, but also because of complexity arising from constraints and dependencies.

Currently, the Requirements Optimisation process, where it is practiced at all, is a highly labour-intensive activity. Search based techniques have the potential to handle large scale problems because they are naturally parallelisable [3,12]. However, despite this potential, there remains a need for more work on scalability of Search Based Requirements Optimisation.

Solution Representation. Visualisation plays an important role in all optimisation problems [9]. It illustrates the solution quality and helps the decision maker to understand the results. This can be easily and directly achieved using scatterplots when there are only 2 or 3 objective dimensions. Visualisation of higher dimensionality remains an open problem in the visualisation community. Requirements Optimisation solutions need to be presented in a manner that is equally intuitive to engineers and to users alike. This represents an additional degree of challenge. There are several visualisation methods for higher dimensional spaces that may be useful, for example Heatmaps [9], Self Organizing Maps (SOM) [8], and Distance and Distribution Charts [1]. However, these remain to be evaluated for Requirements Optimisation.

Feedback and Explanation of Results. In Requirements Optimisation, an additional problem arises when solutions are found: how do developers explain the solution to the customer? Of course, the customer expects to get the highest interest from the solution and they are likely to want to know, not merely the results, but also why a certain set of features was chosen or why some excluded requirements were those for which they had a particular care.

Feedback to the customer should form a part of the solution obtained by the optimisation process. This will maximize each customers' satisfaction and make explicit their participation in the optimisation process. In some cases involving politically sensitive choices, solution justification and explanation may be as important as the solution itself.

Fitness Function Definition. At the heart of SBSE is the fitness function, which guides the search by capturing the properties that make one solution preferable to another. In software engineering applications, fitness functions can be thought of as metrics [6]. These metrics translate constraints such as quality constraints (usability, reliability), organizational constraints (scalability), and environmental constraints (security, privacy) into some measurable attribute of a candidate solution.

Unfortunately, these constraints, often misleadingly termed non-functional requirements, may not be defined precisely in the early stages of the software life

cycle. Therefore, techniques are required for iterative update of fitness function definition. It is possible that fitness-measure and solution-generation may need to co-evolve as part of an overall Requirements Optimisation process.

Algorithm Selection. Search Based Requirements Optimisation is based on experimental results from empirical studies. There is, however, currently little theoretical understanding as to when, how and why one metaheuristic algorithm works better than another. Once the nature of Search Based Requirements Optimisation is better understood empirically, it will be important to generalise these results and to augment them with theoretical analysis of search landscape characteristics. This will support a more formal and rigorous analysis of potential algorithmic complexity, thereby motivating the choice of algorithm to apply.

Requirements Dependencies. In the requirement analysis process, requirements are seldom independent of each other. There are two major problems related to requirement dependencies: one is how to identify and model them, the other is the extent to which these dependencies influence and interact with the software systems level. Ruhe and Omolade [11] show how search based optimisation can track dependencies from user requirements into their impact on system components. Though this is promising, more work is required to handle fuzzy incomplete, multi-way, implicit and temporal requirements dependencies.

Partial Requirement Fulfillment. Requirements have varying representations: *discrete variable requirements* which are either fulfilled completely or not fulfilled at all and *continuous variable requirements* which can be fulfilled to a certain extent, for example sever response time in web-based or distributed systems. Existing work on Search Based Requirements Optimisation has treated requirements as being entirely discrete. More work is required to extend these results to handle continuous requirements.

References

1. Ang, K.H., Chong, G., Li, Y.: Visualization Technique for Analyzing Non-Dominated Set Comparison. In: 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002), Singapore, vol. 1, pp. 36–40 (2002)
2. Bagnall, A.J., Rayward-Smith, V.J., Whittle, I.M.: The Next Release Problem. *Information & Software Technology* 43(14), 883–890 (2001)
3. Collette, Y., Siarry, P.: *Multiobjective Optimization: Principles and Case Studies*. Springer, Heidelberg (2003)
4. Greer, D., Ruhe, G.: Software Release Planning: an Evolutionary and Iterative Approach. *Information & Software Technology* 46(4), 243–253 (2004)
5. Harman, M.: The Current State and Future of Search Based Software Engineering. In: 29th International Conference on Software Engineering (ICSE 2007), Future of Software Engineering (FoSE), pp. 342–357. IEEE Computer Society, Washington (2007)
6. Harman, M., Clark, J.: Metrics are Fitness Functions Too. In: 10th International Software Metrics Symposium (METRICS 2004), pp. 58–69. IEEE Computer Society, Washington (2004)

7. Harman, M., Swift, S., Mahdavi, K.: An Empirical Study of the Robustness of two Module Clustering Fitness Functions. In: International Conference on Genetic and Evolutionary Computation (GECCO 2005), pp. 1029–1036. ACM, New York (2005)
8. Obayashi, S., Sasaki, D.: Visualization and Data Mining of Pareto Solutions using Self-Organizing Map. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 796–809. Springer, Heidelberg (2003)
9. Pryke, A., Mostaghim, S., Nazemi, A.: Heatmap Visualisation of Population Based Multi Objective Algorithms. Technical Report, University of Birmingham (2006)
10. Ren, J.: Sensitivity Analysis in Multi-Objective Next Release Problem and Fairness Analysis in Software Requirements Engineering. Master's thesis, DCS/PSE, King's College London, London (2007)
11. Saliu, M.O., Ruhe, G.: Bi-Objective Release Planning for Evolving Software Systems. In: 6th European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, pp. 105–114. ACM, New York (2007)
12. Szidarovsky, F., Gershon, M.E., Dukstein, L.: Techniques for multiobjective decision making in systems management. Elsevier, New York (1986)
13. Zhang, Y., Harman, M., Mansouri, S.A.: The multi-objective next release problem. In: International Conference on Genetic and Evolutionary Computation (GECCO 2007), pp. 1129–1136. ACM, New York (2007)

Connecting Feature Models and AUTOSAR: An Approach Supporting Requirements Engineering in Automotive Industries

Wolfram Webers, Christer Thörn, and Kurt Sandkuhl

Jönköping University, School of Engineering, P.O. Box 1026
55111 Jönköping, Sweden

{wolfram.webers,christer.thorn,kurt.sandkuhl}@jth.hj.se

Abstract. Due to the AUTOSAR initiative, automotive suppliers as well as their customers and sub-suppliers will in the future face the challenge to exchange AUTOSAR specifications instead of structured documents with arbitrary specification attachments. The consequential main tasks for the suppliers will be to implement model-based representation of artifacts as early as possible in the system development process. Thus, it would be desirable to link strategic elements in the context of product family development and technical aspects concerning model management and representation in AUTOSAR. This paper presents the results of an industrial case study in requirements engineering of software-intensive automotive systems, aiming at a process streamlined to both, the strategic demands of the supplier and the technical demands of AUTOSAR. The main contribution of this paper is the connection of feature models and assets in AUTOSAR descriptions. This paper presents work in progress including the developed concepts and feasibility studies.

Keywords: feature modeling, AUTOSAR, product families, requirements engineering.

1 Introduction

Requirements engineering in the automotive domain is influenced by a number of constraints that are not present in other industries [1]. Some of these are based on the limitations stemming from the system architecture of the networked ECUs found in modern cars. Such networks can consist of up to 80 or more [2] different control units, building up the complex functional car domains, such as safety, comfort, power-train, etc. To tackle this complexity, the automotive industry established the AUTOSAR [3] partnership to standardize both, the future architecture as well as the specification language of such systems. As a result the automotive supplier has to face the challenge the align (1) the new technical demands of AUTOSAR (new tools, new techniques, new methods) and (2) his very own strategic demands for his products (unique market space, specific product, product capabilities to compete, product price, etc) stated among others by his very own market analyses and business goals.

AUTOSAR is developed with respect to the nowadays automotive development process. Systems in the automotive domain are generally evolving in small steps meaning that innovations are included with great care [4]. Consequently, the automotive manufacturer (OEM¹) planning a new system reuses as much as possible from former successfully realized systems. Physical space, electrical power consumption and weight, as the most limiting factors in cars, have a direct impact on the system planning, resulting in so called communication matrices, which basically capture all technical signals with the participating senders and receivers. Given such a matrix for an existing system as a starting point for a new system, a subset can be defined for a specific subsystem, like engine control, airbag system, window lifter, etc. Together with the interfacing specification of the surrounding system, this will result in the tight constraints typically found in the requirements specification for such a subsystem. Thus, the greater part of the requirements provided to the suppliers are more non-functional than functional. For a certain subsystem, like an airbag system, the inner architecture is rather complex, but usually hidden and will not appear explicitly in the customer requirements specifications. These specifications will rather contain more information about the different quality attributes of the demanded subsystem.

Furthermore, car manufacturers develop vehicles in terms of product lines, as described in [2,4,5]. Thus, variability and variance occur on different levels of abstraction during the system planning at the OEM. Some of them will be resolved early by the OEM, but some will propagate down through the whole supplier chain [6]. As the supplier usually has to satisfy more than one customer at once, this manifold will multiply with every customer project the supplier has. Consequently, suppliers need to get an overview of their existing assets as quick as possible during the requirements acquisition process.

In summary the supplier has to face customer requirements inherently having more non-functional than functional requirements one the one hand, and reflecting the car manufacturers' own product line on the other hand. Feature models are widely accepted in the community as a concept for supporting reuse even on the level of requirements. Still, feature models are intermediate models and need to be related to reusable assets and original requirements. Furthermore, they very often focus only on functional aspects. This paper propose an approach to emphasize quality attributes of existing assets in feature models. These quality attributes will be used for supporting the selection of assets required for implementing the specified functionality. As an example for representing existing assets in a detailed descriptive way the approach uses AUTOSAR models.

The paper is structured as follows: Section 2 introduces the background for the work, including an industrial case, the AUTOSAR initiative and feature models. Section 3 presents the approach for connecting feature models and AUTOSAR for supporting requirements engineering in automotive industries. The discussion of our approach, the related and future work (section 4) conclude the paper.

¹ Original Equipment Manufacturer.

2 Background

The background for this paper is the AUTOSAR initiative, in particular the software component specification approach, feature modeling techniques and an industrial case from the automotive supplier industry, which defines the application context. These topics will be briefly introduced in this section.

2.1 AUTOSAR

General Overview. To tackle the complexity of current and future networks of electronic control units (ECU) constituting the different subsystems of modern cars, the AUTOSAR [3] partnership develops an infrastructure for an automotive software architecture. This architecture is illustrated in Fig. 1, showing different software layers consisting of a platform independent application layer, a runtime environment (RTE) on top of standardized interfaces to the middleware (Basic Software), as well as a description of the underlying hardware platform (ECU hardware). The communication between the different components of the architecture relies on standardized interfaces. The idea behind is to support the exchange of components developed by different suppliers as long they support the according interface.

AUTOSAR specifications are expressed in a modeling language, based on a hierarchical UML meta-model. The language covers different areas of concern and is organized in so-called AUTOSAR template definitions. Each template

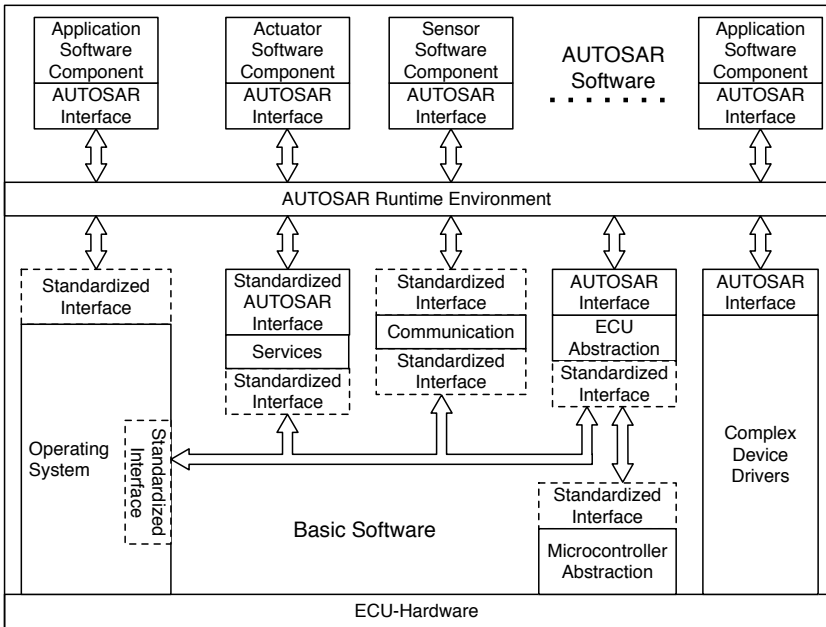


Fig. 1. AUTOSAR Architecture Overview (adapted from [7])

relates to specific aspects of the architecture, namely application software, ECU-hardware and system topology. This approach provides both the support of existing modeling tools as well as a standardized, XML-based exchange format.

Additionally, these three AUTOSAR templates cover the different notions typically found in the requirements specifications in the automotive domain [2]. Requirements of functional aspects, system communication aspects (e.g. communication matrices) and hardware aspects can utilize the modeling facilities of the corresponding AUTOSAR template. The approach presented in this paper is currently limited to software components on the application level. Thus, we assume the supplier is responsible for developing the application part of the architecture. Other scenarios are also possible, and need further investigation. Furthermore, this paper focuses only on the structural aspects of the application software. The behavioral aspects are intentionally left out.

AUTOSAR Software Components. Software components in AUTOSAR can be described on three different levels of abstraction as depicted in Figure 2. The most abstract level describes the so-called Virtual Function Bus (VFB), which focuses only on the abstract communication aspects. Thus, it captures *who* is present and is communicating with *whom*.

Software applications on this level are described by subsystems made of components with ports and interfaces. Logical hierarchies of software components can be built up by compositions. Atomic software components contain functionalities (Runnables) of the application running on the ECU which will be part of the scheduling. Such components cannot be decomposed further, therefore the name.

The communication between software components differs only between the synchronous and asynchronous communication pattern and is completely captured with the concepts of ports, interfaces and connections. Furthermore, the

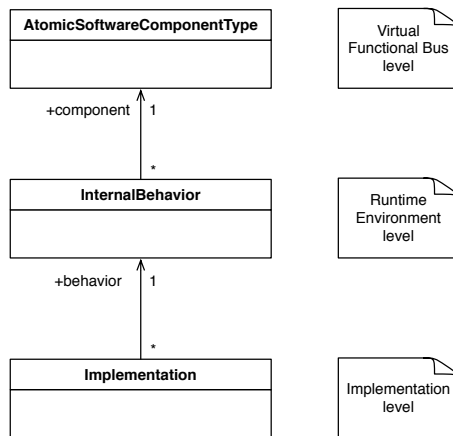


Fig. 2. Three Levels of Software Component Descriptions (from [8])

communication is abstracted from the underlying hardware. Thus, technical signals are abstracted to logical representations in the form of data types.

The second abstraction level defines the internal behavior of software components and the communication with the runtime environment. The communication with the RTE is described by means of events, schedulable units (Runnable Entities), as well as schedulability aspects. Communication with hardware elements is completely routed through the runtime environment (RTE), but it has to be represented on the application level as well. Thus, the communication with sensors is done by using special derivatives of atomic software components on the application layer representing those sensors.

The lowest abstraction level finally describes the implementation of such internal behavior by mapping them to code, either source or binary. The implementation can further be annotated by several quality attributes, like resource consumption or runtime estimations. In general, this level describes the how of the components realization.

2.2 Feature Modeling

The concept of feature models was originally introduced in 1990 by Kang et al. with the FODA technical report [9]. Originally used for domain analysis in the telecom domain, feature models are now used in other domains and for other purposes. The original definitions, notations and concepts used by Kang has been extended and modified over the years to fit new uses. Feature modeling is today a technique that is incorporated in several software and systems development methodologies.

The purpose of feature models is to extract, structure and visualize the commonality and variability of a domain or set of products. The variability and commonality is modeled as features and organized into a hierarchy of features and subfeatures, often called a feature tree, which is usually visualized in a graphical description of the commonality and variability found in the modeled domain or part of domain.

Between the features in the model, there are relations describing the constraints of the features' composition possibilities. The feature/subfeature hierarchy describe the principal restrictions on how features in a product family can be combined in an instance of a product line, but there are further restrictions that pose constraints on what parts of a product line that can be combined and configured. These restrictions could create dependencies or exclusions across different branches of the feature tree and reach considerable complexity. The feature tree visualize those dependencies and connections between features and the subsumption hierarchy of features and subfeatures, so that the engineer can utilize the variability in product engineering or other configuration activities.

Definitions of Features. There are several definitions of the term feature used in conjunction with feature models. Some of them are more formal, while others are more intuitive. Features are intended to be concepts described by a single

word or short line of text. We select a few definitions from other authors that are aligned to the idea of this paper:

- From FODA by Kang et al: “A *prominent or distinctive and user-visible aspect, quality, or characteristic of a software system or systems.*” [9]
- From IEEE: “A *software characteristic specified or implied by requirements documentation (for example, functionality, performance, attributes, or design constraints).*” [10]
- From Riebisch et al: “A *feature represents an aspect valuable to the customer.*” [11]

While many definitions and usages of features and feature models are directed towards functional characteristics, our work also consider features aimed to non-functional characteristics and quality attributes. More specifically, we consider features representing functionality to have attributes of quality characteristics. Thus, our approach utilizes an extended feature model similar to the one presented in [12] and shown in Fig. 3. The feature model presented in [13] is extended in a similar way.

Family Models. Feature models usually do not exist alone, but are related to reusable assets describing the solution space. Such assets need to be represented in a manner so that they can be configured and merged to final single solutions. This raises specific requirements towards the notation of the assets specification. For example, assets must be described in a way they can be composed or merged to final products. The CONSUL approach in [13] defines a component based family model for this and uses logical constraints to relate existing solutions to features in the feature model. This approach follows a process where a concrete selection of features defines a product configuration. Such a configuration consists

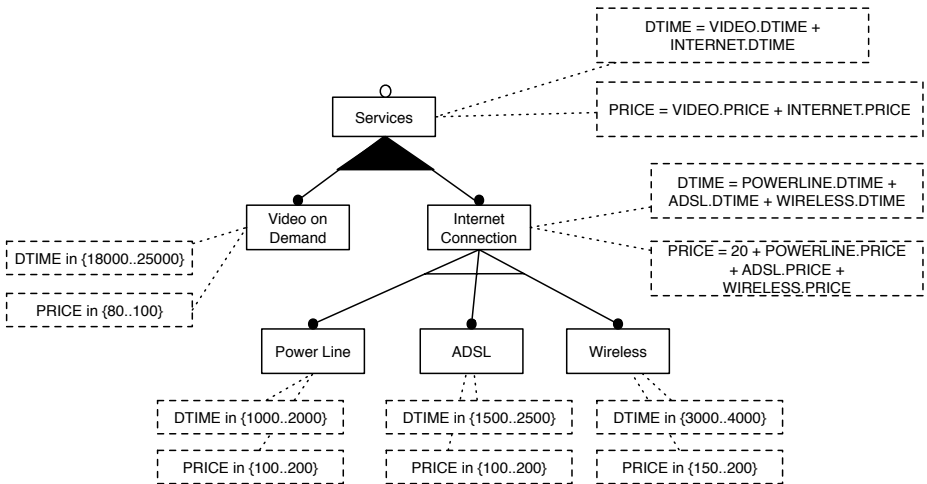


Fig. 3. Extended Feature Model (from [12])

of the parts from the product family which fulfill certain constraints attached to them and are merged together to a single product.

In CONSUL, family models compose configurable sets of functionalities in hierarchies defined by logical components and parts. Components can have an arbitrary number of parts. Parts collect concrete solutions by referencing to specific sources of solutions. Those solutions can be of arbitrary types like classes, objects or source code.

Additionally, CONSUL applies a process of deriving single solutions out of the family model by creating configurations of features. Those configurations consist of a subset of the original feature model by selecting those features to be realized in a single solution. To be able to derive a single product the family model needs to be formally related to the feature model. In CONSUL this is done by several logical expressions which are evaluated during the configuration process. For every asset these expressions need to evaluate to true to be a valid member of the solution.

2.3 Industrial Case

The industrial background for this paper is a Swedish automotive supplier of software-intensive systems. In the research project SEMCO, different aspects of the systems development process of this automotive supplier were studied with focus on requirements engineering. The major focus of the project was to create an enterprise ontology for supporting the requirements engineering process by applying ontology matching methods [14]. The idea was to have in the first step both, the original requirements specification and existing asset specifications described with the help of ontologies [15]. As ontologies themselves are rather general concepts, we additionally mapped those to the concepts of feature models [16] with the result of having feature models as subsets of an overall enterprise ontology. In the following we focus only on the aspects belonging to the feature model and AUTOSAR concept of the project.

During this project, a feature model for the product line 'Airbag Control Systems' of the automotive supplier was developed. As the SEMCO project aimed at supporting requirements engineering, the intention was to develop the feature model from this perspective with the intention to match incoming customer requirements as fast as possible to existing solutions. On this basis two system specification documents of existing airbag solutions developed for different customers were analyzed in order to find visible features of this product line. As a result the functionalities of airbag controllers were captured in a feature model representing the two solutions in one model.

The feature model was complemented by a family model. The family model should show all the existing solutions the automotive supplier has for building airbag systems. Based on the provided specifications it was decided to define two family models: one for collecting all the hardware parts, and one for the software parts of an airbag system. The family models consist of hierarchical trees containing components like they are built by following the CONSUL approach.

One challenge was to match the resulting feature model against the original customer requirements. As mentioned above those requirements state less functionalities than non-functional aspects. Additionally, they list up to a larger extent hardware components to be used in the solutions. The other challenge was to find a link between the feature model and AUTOSAR specifications as the family models defined in the CONSUL approach need explicit relations to the reused assets. In the following we describe our approach to tackle these challenges.

3 AUTOSAR Software-Components as Family Models

The approach proposed here concerns the requirements engineering process on the supplier' side of the OEM-supplier collaboration. Within this requirements engineering process, the goal is to support the identification of existing development assets fitting to certain demands stated in the customers requirements specifications. The AUTOSAR initiative aims at a process where the customer requirement specifications to a large extent is expressed as models, i.e. our approach assumes at least partial AUTOSAR specifications as input from the OEM to the supplier.

The approach uses the concept of feature models as a mediator between those requirements and existing assets. As an organization develops and markets a range of products, it creates various technical solutions that satisfy the requirements posed on the products by the customers. These solutions represent variability in the capabilities of the product line and potential alternatives for use in development of new products. By structuring the products' current abilities via feature models, the various configurations that are available in the solution space can be modelled with respect to the requirements. Thus, the feature models describe the problem space or requirements space of the product line, while the solution space is represented by corresponding family models, which model the core assets that are available to realize the instances of the product line. In our approach, feature models are used for two purposes:

- To express specific capabilities of existing assets with AUTOSAR techniques. The construction of such a feature model is guided by the mapping presented in section 3.1.
- Matching the customer requirements to existing assets. The same mapping rules will be used here and the feature model further guides the decision process in selecting existing assets with respect to their specific capabilities.

In our case, the solution space is populated by AUTOSAR specifications of software components, which represent the various available assets. The feature model links the solutions to the requirements posed on the product to be derived from the product line.

3.1 Mapping between Family and Feature Model Concepts

The approach presented in this paper aims at using AUTOSAR facilities provided by the software component template and relates those to the family model in the CONSUL approach.

Fig. 4 shows a more detailed overview of the elements taken from the software component template and used in the approach. As mentioned above, compositions are used for structural reasons and do not influence the concrete realization of the application. Thus, we can identify AUTOSAR components with the component concept in the CONSUL approach. Components in AUTOSAR, and therefore all possible descendants of atomic software components, are containers for the concrete realizations. Those will be identified with the parts concept in CONSUL. The internal behavior is the container for both the possible communication offered by the runtime environment (exemplified with *RTEEvent*) and the schedulable parts (*RunnableEntities*). Finally, the implemented solution (*Code*) together with several quality attributes (exemplified by *StackUsage* and *ExecutionTime*) resides in a common container (*Implementation*). It indirectly relates to the software component whose functionality is realized by the code.

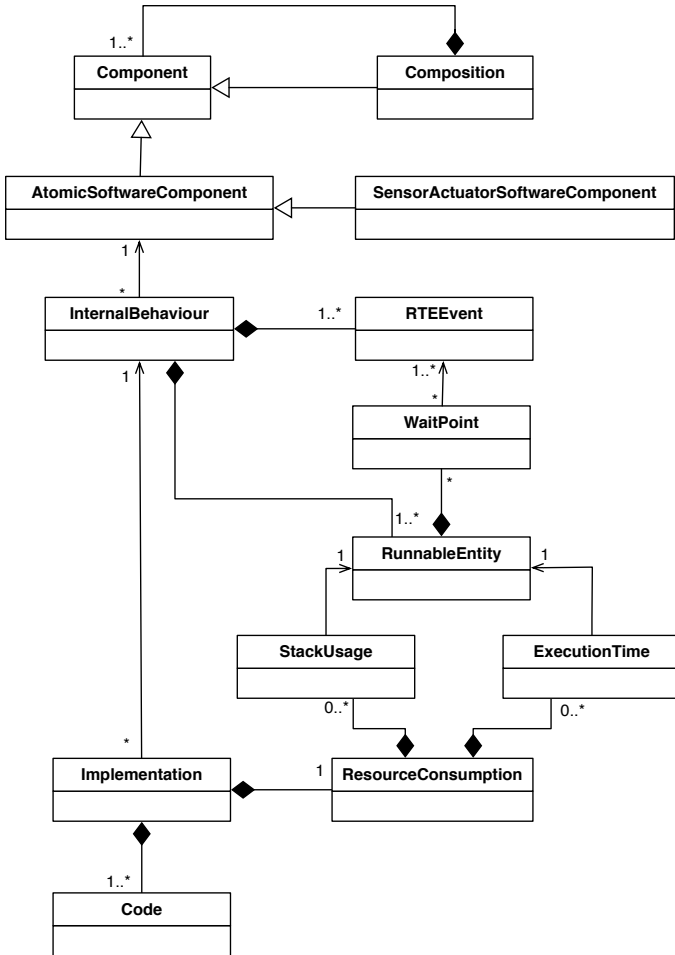


Fig. 4. Software Template Elements used in our Approach

Table 1. Mapping between CONSUL and AUTOSAR

	CONSUL	AUTOSAR
Feature	Feature	AtomicSoftwareComponents Composition
Feature	FeatureAttribute	ResourceConsumption
Family	Component	Composition
Family	Part	Component InternalBehaviour
Family	Source	Implementation

In this paper we leave out further details about the communication parts (ports, interfaces and connections), as well as hardware resources. Table 1 summarizes the mapping of the concepts found in CONSUL and AUTOSAR.

For the feature model we have chosen the concepts of atomic software components and compositions as targets for the mapping. This was done due to the more "component-focused" original requirements specifications found in the case study. We additionally annotated these features with the concept of resource consumptions as feature attributes. The concrete values for these attributes come directly from the non-functional requirements found in the specifications. With such mapping we were able to describe both, the problem space with the help of the feature model and the solution space with the help of AUTOSAR.

3.2 Example of the Quality Driven Feature Resolution

The following example, abbreviated for confidentiality reasons, provided by our case study describes the problem of matching a solution for a restraint algorithm of an airbag system. Both models are built by applying the mapping described before. The "stereotypes" in the models are used for clarifying the role of the model elements.

We assume that the customer requirements contain a statement about the quality of such a system, defining specific timing requirements and memory resource consumption. In Fig. 5 an excerpt of the feature model is given, showing the main feature for the algorithm as well as the exposed quality features for this feature. The concrete quality demands, coming from the requirements specification, are instantiated as feature attribute values. The family model expressing the existing solutions is shown in Fig. 6. This example contains two solutions for the demanded functionality with different realizations and quality attributes.

The relation to the feature model is built up by the logical constraints annotating the components and parts in the family model. Our approach is therefore following the same approach presented with CONSUL. The feature matching is done in the traditional way by matching the names. In our case we match component names. Additionally, the deeper matching is driven by the concrete quality attribute values stated in the customer requirements. This will decide on the concrete implementation of the component (if such exist). For the case such

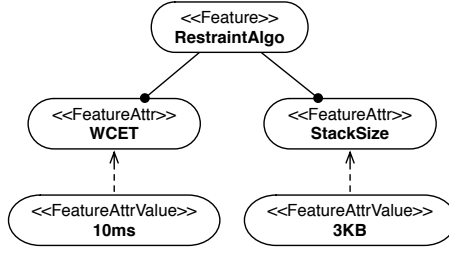


Fig. 5. Feature Model for Restraint Algorithms

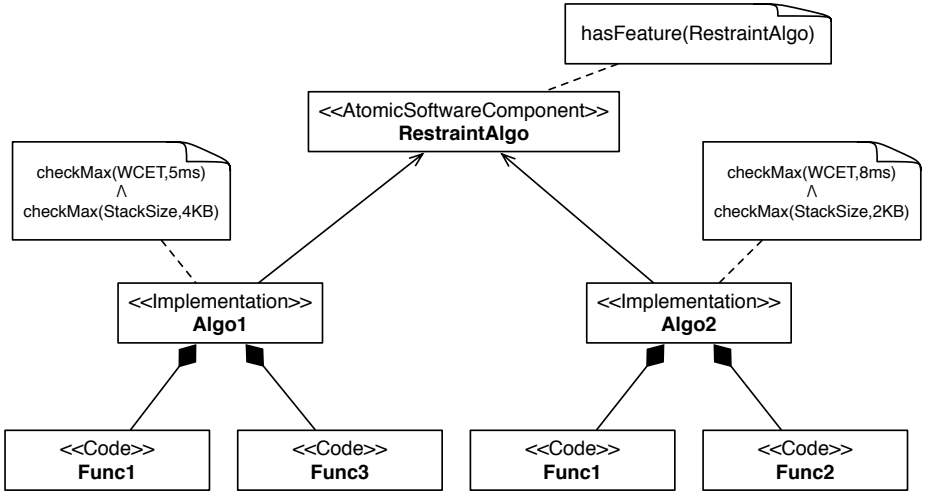


Fig. 6. Family Model for Restraint Algorithms

constraint evaluation is successful the matching implementation can be seen as potentially reusable solution.

For the example feature model, when selecting a restraint algorithm in a concrete configuration, only the second algorithm fulfills the formal constraints. Thus, the development process can potentially reuse this best fitting existing implementation.

4 Discussion and Future Work

This paper presents concepts and constructs for connecting quality attributes expressed in feature models to AUTOSAR specification assets. It addresses the need for variability and variance management on different abstraction levels in an automotive-oriented requirements acquisition process, with a special focus on quality attributes.

We recognize non-functional requirements as important parts of any customer requirements specification in the automotive domain. The AUTOSAR meta-model offers a detailed model-based description of those entities. Having AUTOSAR descriptions of existing assets by hand makes it easier to drive a decision process driven by quality attributes, as it is exemplified in this paper.

We see AUTOSAR as an important and highly relevant factor for the development of embedded automotive software. It enables development processes and in particular the requirements management to leverage existing assets in the form of AUTOSAR specifications. For suppliers in the automotive domain with extensive existing assets, the development for different needs and customers using compliant specifications is a decisive challenge. Streamlining the activities between requirements engineering and the main product development is an important task.

4.1 Related Work

Using feature models to ease or solve the problems of embedded software development for the automotive industry has been treated by several authors and research groups, predominantly in Europe. The survey in [17] gives an overview on the work done in this area. In contrast to the question treated in [17] about variability found on the level of quality attributes, the approach presented in this paper has its focus on matching customer requirements to existing solutions in a product family by using quality attributes.

Czarnecki et al. describe using attributes for features in the development of embedded systems, applicable to the automotive domain [18]. The problems of modeling variability and the peculiarities of software product lines for the automotive domain has been covered in for instance [19].

The work conducted in AUTOSAR is often mentioned and referenced as a relevant and important effort that will significantly influence the work flows and development aspects in the future [2]. To date, most of what has been written about AUTOSAR has been speculative when it comes to a concrete application of AUTOSAR, pending the realization by the initiative. Our work contributes a concrete merger of feature-driven development using product lines and AUTOSAR-compatible constructs.

4.2 Future Work

The approach presented here focuses on the scenario where the supplier delivers the application part of the software. Our work utilizes the atomic software component concept, focusing on the variability of the software implementations depending only on quality attributes and exclude the communication and hardware resource dependencies. Other scenarios concerning the basic software parts, the hardware parts and combinations of those, should also be considered. The approach presented here can be extended in order to cover all quality attributes found in the AUTOSAR meta-models.

Acknowledgements. Parts of this work are financed by the Swedish Knowledge Foundation (KK-Stiftelsen), grant 2003/0241, project SEMCO.

References

1. Pretschner, A., Broy, M., Kruger, I.H., Stauner, T.: Software engineering for automotive systems: A roadmap. In: FOSE 2007: 2007 Future of Software Engineering, Washington, DC, USA, pp. 55–71. IEEE Computer Society, Los Alamitos (2007)
2. Becker, M., Böckmann, C., Kamsties, E., Wierczoch, T.: Requirements engineering in the automotive development: Known problems in a new complexity. In: 12th International Workshop on Requirements Engineering Foundations for Software Quality (REFSQ 2006), Luxembourg (2006)
3. AUTOSAR: AUTOSAR – Automotive Open System Architecture (2007), <http://www.autosar.org/>
4. Bittner, M., Botorabi, A., Poth, A., Reiser, M.O., Weber, M.: Managing variability and reuse of features and requirements for large and complex organizational structures. *re* 0, 469–470 (2005)
5. Bühne, S., Lauenroth, K., Pohl, K., Weber, M.: Modeling features for multi-criteria product-lines in the automotive industry. *IEE Seminar Digests* 2004(914), 9–16 (2004)
6. Czarnecki, K., Helsen, S., Eisenecker, U.: Staged configuration through specialization and multilevel configuration of feature models. *Software Process: Improvement and Practice* 10(2), 143–169 (2005)
7. AUTOSAR: AUTOSAR – Technical Overview. Version 2.0.1 (2007)
8. AUTOSAR: AUTOSAR – Software Component Template. Version 2.0.1 (2007)
9. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR-21, ADA235785, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA (November 1990)
10. IEEE Standards Board: IEEE Standard Glossary of Software Engineering Terminology. Technical Report IEEE Std 610.121990, IEEE (1990)
11. Riebisch, M.: Towards a more precise definition of feature models. In: Riebisch, M., Coplien, J.O., Streitferdt, D. (eds.) ECOOP 2003. LNCS, vol. 2743, pp. 64–76. Springer, Heidelberg (2003)
12. Benavides, D., Trinidad, P., Ruiz-Cortés, A.: Automated reasoning on feature models. *Advanced Information Systems Engineering*, 491–503 (2005)
13. Beuche, D., Papajewski, H., Schroder-Preikschat, W.: Variability management with feature models; software variability management. *Science of Computer Programming* 53(3), 333–352 (2004)
14. Blomqvist, E., Öhgren, A., Sandkuhl, K.: Ontology construction in an enterprise context: Comparing and evaluating two approaches. In: Manolopoulos, Y., Filipe, J., Constantopoulos, P., Cordeiro, J. (eds.) ICEIS, vol. (3), pp. 86–93 (2006)
15. Sandkuhl, K., Billig, A.: Ontology-based artefact management in automotive electronics. *International Journal of Computer Integrated Manufacturing* 20(7), 627–638 (2007)
16. Sandkuhl, K., Thörn, C., Webers, W.: Enterprise ontology and feature model integration: Approach and experiences from an industrial case. In: ICSoft 2007: Second International Conference on Software and Data Technologies (May 2007)

17. Etxeberria, L., Sagardui, G., Belategi, L.: Modelling variation in quality attributes. In: Pohl, K., Heymans, P., Kang, K.C., Metzger, A. (eds.) VaMoS: First Intl. Workshop on Variability Modelling of Software-intensive Systems. Technical Report 2007-01, Lero (January 2007)
18. Czarnecki, K., Bednasch, T., Unger, P., Eisenecker, U.W.: Generative programming for embedded software: An industrial experience report. In: GPCE 2002: Proceedings of the 1st ACM SIGPLAN/SIGSOFT conference on Generative Programming and Component Engineering, London, UK, pp. 156–172. Springer (2002)
19. Thiel, S., Hein, A.: Modeling and using product line variability in automotive systems. *IEEE Software* 19(4), 66–72 (2002)

Using a Creativity Workshop to Generate Requirements for an Event Database Application

Claudia Schlosser, Sara Jones, and Neil Maiden

ComNetMedia AG, Emil-Figge-Str.86, D-44227 Dortmund, Germany
Centre for HCI Design, City University, London, EC1V 0HB, UK
s.v.jones@city.ac.uk

Abstract. This paper describes one experience of using a creativity workshop to generate requirements for an event database application for a network of German Chambers of Commerce (CCI's). The workshop described was the first to be run by the host organization. Techniques used during the workshop included discussion of system boundaries and use of creativity triggers. We discuss the results from the workshop in terms of the number and importance to stakeholders of the requirements generated. We end with a presentation of lessons learnt for improved creative practices in requirements engineering.

Keywords: requirements acquisition, creativity workshop.

1 Introduction

The role of creative thinking in requirements engineering has been recognized as important [1], [2], but creativity techniques have yet to be employed widely in requirements projects. In this paper we report the application of published requirements creativity techniques during a workshop in a project developing an event database application for German Chambers of Commerce. Both the project and the organization co-ordinating it were smaller than in other reported applications of similar techniques (see, for example, [1], [2] and [6]). Results reveal the positive impact of the workshop on participants in the requirements process, and the relative effectiveness of some of the creativity techniques.

The remainder of the paper is in 4 sections. Section 2 describes the organizational context within which the work took place. Section 3 describes the project in which the workshop was applied, and section 4 describes the workshop itself. Section 5 reports results from the workshop and post-workshop analyses of generated requirements. The paper ends with a discussion of lessons learned for the organization and for creative requirements engineering activities in general.

2 Requirements Engineering at ComNetMedia

ComNetMedia (CNM) is an IT solution provider, which was founded in 2000 as a spin-off of the CCI Gesellschaft für Informationsverarbeitung mbH. It is responsible

for about 200 different applications used by German Chambers of Commerce (CCIs), including databases, enterprise content management systems, archive solutions and email. In most cases the applications have an interface to the database of the majority of the German CCI organisation (integrated systems), and need to conform to e-government standards, which has a significant impact on IT architectures and business processes.

CNM has two main branches – development and consulting. The consulting branch consists of 10 project managers / consultants and is responsible for the RE process as interface between customers and developers. The senior consultants and team leaders have nearly 10 years experience in requirements engineering. CNM often carries out requirements work on behalf of the chamber organization, with development being carried out by CNM or another company. RE is therefore one of CNM's core business areas.

Depending on the project type and size, the RE process is adapted as described in the internal CNM project handbook. For projects of all sizes, the handbook contains models and examples of requirements descriptions, and a description of the internal CNM process for requirements management. For medium and large size projects, CNM uses Quickplace, a LotusNotes based Groupware tool to give transparency and management of RE functions such as change requests, description of work as use cases, and incorporation of RE into project plans. Other tools are available for RE description and management, including style guides relating to GUI design, tools for ER diagrams, use case and process modelling tools.

A new project, to build a new application, starts in most cases with some fuzzy ideas and requirements from the customer or CNM. Analysts at CNM then start to identify additional requirements. After meetings with customer representatives, or further partners, and CNM developers, CNM starts to write the technical concept (use cases, uml etc.) and discuss the results with customers, CNM developers and management. These steps are performed iteratively. However, representatives are often high-level people from the customer organisations, and access to the real target group, which will use the application in their workplace, can be difficult. This problem is reflected in the meetings with customers, when the selected group of representatives is not involved in the daily business process and has no detailed experience with the real issues for users. It means that many requirements are often detected relatively late in the project, when a prototype is in place and more users have access to it. In most cases the new requirements are not “cost neutral”. In these cases, change requests are collected and evaluated by CNM in terms of feasibility, budget, and delivery (release). Then the customer can decide whether the importance/use of the new “functionality” is at least equal to the additional budget/effort.

One horizontal task within CNM RE work is to improve methods to deliver a higher quality RE process and ultimately a better quality of finally product or project. In this paper, we describe the experiences of one of the authors, who is a senior consultant within CNM, of using a creativity workshop as part of the CNM requirements process. This author had first encountered creativity workshops in the context of the APOSDLE project [3], first by participating as a stakeholder in one such workshop, and then learning more about them from the APOSDLE work-based learning prototype [4]. The idea of running creativity workshops as part of the requirements process was initially developed in the context of the RESCUE requirements process [5].

RESCUE creativity workshops have much in common with other participatory design workshops, but are designed specifically to stimulate creativity, using established models of creativity from artificial intelligence and social and cognitive psychology, as described in, for example [6]. Creativity workshops sit between the four streams of the RESCUE process, drawing input from early models of actors and use cases for the future system, and providing output which is used in particular to help specify use cases and identify requirements for the future system. Outputs from such workshops include requirements, creative design ideas, and storyboards embodying the creative ideas inspired by the workshop. These are used by those who write use cases and requirements as part of the future system specification. Workshops are designed based on models of creativity from cognitive and social psychology, as described in [6] and normally run for two days, incorporating a number of different activities designed to stimulate creativity. The RESCUE team has so far facilitated 14 creativity workshops in the air traffic management, policing and self-directed learning domains (see, for example, [1], [2]). However, RESCUE-style creativity workshops have not previously been run by facilitators from outside of the RESCUE team. This paper describes the first occasion on which this has been attempted.

3 The Event Database Application (EDA) Project

The project in which CNM decided to trial the use of a creativity was to develop an event database application for German CCIs. The first version of the application was built using a content management system. Further development was then transferred into an internal knowledge management project of the DCCI: the association of the German CCIs. As part of this project, “event publication and management” was defined as a sub-system and realised as a database application. Several requirements, such as interfaces to other systems, XML-import/export functions, event management etc. were realised. This version of the application is currently used by German CCIs for offering and “booking” events and training sessions. There are several types of events, such as free-of-charge information events; expensive long term training courses, including examinations, that can lead to degree level qualifications; one day basic training courses about, for example, “how to use MS Word”; or workshops on how to set up a new company.

Three years ago there was a platform change resulting from the fact that the old versions were no longer supported by the software and hardware. The application was updated several times so that the technical platform was “state-of-the-art” but the application (business logic) itself was not. The “old-fashioned” event database application needed an update. At this stage a decision had to be made: whether to simply change the platform again and keep the old concepts, or to take the chance to start from scratch and develop a new concept and IT architecture. CNM management, together with developers and the EDA project leader decided to start from scratch and build a new application, which would be appealing to users, with additional features and modules in a new architecture. It was decided that input from users and customers should be an important source of information for the new concept. Since, as described above, staff at CNM had access to information about the RESCUE user-centred requirements process, due to their participation in the APOSDLE project [3], it was

decided to use one of the techniques described within the RESCUE process – a creativity workshop -- to obtain inputs from the CCI user group.

Several target groups were identified, including external customers interested in CCI training courses, and CCI staff involved in training and event management, marketing, administration, and overall management and control. Holding the creativity workshop as a “live event”, rather than simply consulting experts or writing down concepts, seemed to be a good start to get a wide range of the different target groups from different CCIs together. The aim was to collect ideas from the different target groups in a “democratic” way and not only to ask some experts or write the concept without asking users. The techniques chosen were intended to support the creative invention of requirements from heterogeneous, non-technical user groups, and the structuring of those requirements around key use cases.

4 The EDA Project Creativity Workshop

18 CCI representatives responded to the invitation to the workshop, including project leaders responsible for CCI web sites and training and course management. Participants came from 12 different CCI's within the CCI24 group, which consists of nearly 30 chambers of commerce [7]. All of these chambers have the old system in place. In addition, to add some more technical expertise on systems and tools running in different CCI's, 2 senior consultants, the CCI24 project leader and a trainee from within CNM were also invited.

The representatives from CCI's were well prepared. As part of the invitation to attend the workshop, they were asked to be prepared with detailed knowledge of CCI internal processes (e.g. how to proceed with the application process for a training course, editing of events in the application, types of events etc.) and experience with the existing application. In most cases CCI representatives collected some feedback from their colleagues and brought lists of ideas (problems) to the workshop. The facilitator spent one day preparing for the workshop, and 4 hours, with the help of some technical support, preparing the space in which the workshop was to be held.

The workshop lasted for 8 hours, with 45 minutes break for lunch. It was held in a large meeting room in Dortmund, and was facilitated by the first author. A first draft of the system context model and use case precis (unstructured paragraphs describing the behaviour of actors in a potential use case) provided the structure for the workshop room itself. The credo was “no ideas off limits” – think of anything, which might be a good idea or should be prevented. Participants were told that all requirements they would identify might be realised in the new application, but that the evaluation of this would be done by CNM, since they were the solution owners, and had an overview of existing and currently planned IT architectures and applications, allowing them to exploit possible synergies with other applications. The workshop was facilitated to encourage a fun atmosphere so that the stakeholders were relaxed to generate and voice ideas without fear of criticism. During creativity periods, standard RAD/JAD facilitation techniques and rules [1] such as avoiding criticism of other people's ideas and time-boxing each topic under discussion were applied. Stakeholders were supplied with Volere requirement shells [9], print-outs from the current application, A3/4 paper, color pens, pencils etc. with which to record the results from each period.



Fig. 1. Scene from the creativity workshop

4.1 Pros and Cons of the Current Situation

The morning period activities began with two ‘round-robin’ sessions in which each stakeholder was asked to come up with features or ideas for the new system based on their experience with the existing application. Participants were given approximately 5 minutes, working alone, to identify the disadvantages of the current system, and half an hour was then allowed for each participant to tell the result to the group. The same procedure was performed for the advantages of the system. The aim of this session was to allow participants to concentrate on the current limitations and identify weaknesses and strengths of the current version. But it was also to get their own favorite ideas or important features out into the workshop up-front, so that they would not use time in subsequent sessions trying to get those ideas heard. Participants were allowed to contribute more than one idea each.

4.2 Definition of System Boundaries

The morning period activities continued with system-wide brainstorming and the identification of system boundaries, considering other systems used within different CCIs where different direct connections for import and export of data are in place. This led to constraint and boundary identification and cleared up the focus and scope of the future development, and of course of the expectations of the workshop day.

The session began with a prepared flip chart showing the first draft system context model, where the system was in the centre and two “rings” around it defined the different layers: the user front-end and GUI; any co-operative adjacent systems [9]; and autonomous adjacent systems. To get the discussion and idea flow started, participants reviewed the “general story” of the application from the point of view of different target groups including the customer, the CCI, the system itself, and external

systems. To drive the session forward, the facilitator then asked open questions e.g. “who or what is part of the application process”. Questions focussed on connections to other systems or actors, connections to different departments in the CCIs (different actors / roles within the CCI organisation), connections between the customer and the system, relation(s) between actors, and relations between use cases or functions and external actors (systems, humans, regulations etc.).

For each activity, actor or system mentioned, assistants added different coloured and shaped post-it notes onto the chart. In addition the group started to create connections between them. The session finished by considering the main use case precis. The aim of this session was to generate a common understanding of what was in and out of scope. This was essential as a lot of the participants’ initial concerns had been to do with external systems.

4.3 Using Creativity Triggers to Generate New Requirements

Two sessions during the workshop were dedicated to generating new requirements using exploratory creativity stimulated by the use of creativity triggers. In each of these sessions, participants were divided into four groups with four or five representatives from CCIs and one from CNM. The moderator created the groups in a way that people from different CCIs and departments, and with different experience (as marketing experts, technical experts or event managers) worked in groups together. The aim was to have groups which brought individuals with different expertise and focus together to prevent “specialisation”. Groups worked in parallel, using different creativity triggers.

The creativity triggers used were those defined in [10], and were explained by the facilitator using the context-relevant examples shown in Table 1. Groups were able to choose which trigger they wanted to work with during each session. Each group worked on using its chosen trigger to identify requirements for approximately 30 minutes, documenting new requirements using the Volere requirement shell [9] translated into German. During this time, the facilitator was available to answer questions if needed, but did not otherwise intervene. After each round each group presented their ideas to the workshop as a whole. This often led to the identification of further requirements. After each round, the participants were re-grouped and chose a new trigger for the next round.

After the workshop, all the identified requirements were recorded in an MS Excel spreadsheet. The CNM project manager structured the list of requirements by relating them to a rough cluster of basic use cases and identifying those requirements which could be used within different use cases (e.g. print, e-mail reminder) as system-level requirements. The spreadsheet was then placed on the CCI24 partner server. This allowed responsible CCI stakeholders, who were not able to participate in the workshop, to be informed and provide additional ideas to CNM. Several new requirements were identified in this way. Finally, all CCI24 project leader participants were asked to rank the requirements, using the Volere satisfaction and dissatisfaction rating scales [9], on behalf of their CCI. This feedback was then collected and used for our internal ranking.

Table 1. Creativity triggers and EDA-specific examples used for explanation in the workshop

Creativity triggers	Context-specific explanations and examples used in the EDA workshop
Service	Target group: customer Target group: CCI event/course management Target group: CCI training Target group: CCI public relation and others
Information	Which kind of information is interesting for customers? Which kind of information could a CCI offer? Which kind of information is useful for the customer? Which kind of information is useful for the CCI?
Participation	How can customers actively participate? How can CCI training course representatives actively participate? How can CCI event management people actively participate? How can CCI PR people actively participate?
Connections	Media for customers Connection to ECMS Connection/ Interfaces to other CCI systems Connection of further media (information) channels/systems
Trust	Customer point of view - System Customer point of view - CCI CCI System
Convenience	Customers CCI course/event management CCI training department CCI PR, communication

5 Results and Discussion

During the workshop, a total of 148 requirements were generated. 34 requirements had been identified by participants in preparation for the workshop, and a further 5 were identified by the facilitators on immediate reflection after the workshop. In this section, we analyse data relating to the 148 requirements generated during the course of the workshop to answer a number of research questions of interest. The main outcomes are shown in Table 2.

Table 2. Numbers of requirements generated from the initial round robin session and the use of different creativity triggers during idea generation sessions

Technique/Creativity trigger	No. reqts
Round robin pros & cons	41
Service	33
Information	25
Participation	0
Connections	4
Trust	9
Convenience	36
Total	148

5.1 What Triggers Did Groups Choose to Work with?

During the idea generation sessions, groups were free to choose which creativity triggers to work with. Table 3 shows how many times a group chose to work with each of the available triggers.

It is interesting to note the differences in the numbers of groups opting to work with the different creativity triggers. Triggers are shown in the table in the order in which they were explained during the workshop. Therefore, the differences may be due to a combination of recency and primacy effects, whereby participants remembered better the earlier and later triggers from the list. However, the impression of the facilitator was that some triggers did not seem as relevant as others, and were not so easy to understand for the participants in this workshop. For example, the ‘Connections’ trigger was explained as quite a technical concept, relating to interfaces with other CCI systems, and may therefore not have seemed very relevant to the stakeholders’ perceptions of the system in terms of its user interface. Further investigation of this issue is needed.

Table 3. Numbers of groups who chose to work with different creativity triggers

Creativity trigger	No. of groups
Service	4
Information	3
Participation	0
Connections	1
Trust	2
Convenience	4

5.2 How Productive Were the Different Techniques Used during the Workshop?

In Table 4, we present a measure of the relative productivity of the different techniques and triggers used during the idea generation sessions. The data shown was generated according to the formula:

$$\frac{\text{number of requirements generated during the session}}{(\text{total number of minutes in the session} \times \text{number of repetitions of session} \times \text{total number of people involved in the session})} \quad (1)$$

This measure is intended to give an approximate representation of the number of requirements generated per person-minute. Note that this is only an approximate measure, since sessions lengths are approximate (correct to within + or – 5 minutes), and group sizes for the idea generation sessions were sometimes 5 and sometimes 6 (an average of 5.5 was used for the calculations).

It is interesting that the round robin session, involving all participants, appears less productive than the work with some of the creativity triggers, which was done by smaller groups working in parallel, although it should be remembered that this session served other important purposes in terms of allowing participants to share ideas and build a common sense of purpose.

Table 4. Productivity of different techniques and triggers

Technique/Creativity trigger	No. of requirements per person-minute
Round robin pros & cons	0.027
Service	0.050
Information	0.051
Participation	N/A
Connections	0.024
Trust	0.027
Convenience	0.055

Looking at Tables 3 and 4, it is also interesting to note an apparent correlation between the popularity of the creativity triggers (i.e. how often they were chosen by groups) and their productivity, with Service, Information and Convenience being the three most popular triggers (chosen by 3 or 4 groups) and apparently also the most productive (with a productivity measure of 0.05 or more). Both choice of trigger and productivity in working with a trigger are likely to be indicators of how meaningful different triggers are to stakeholders with particular experience in a particular domain. These results therefore lend support to the hypothesis that certain triggers may be more meaningful to participants working in particular domains than others. Again, further research is needed to investigate this.

5.3 Does the Use of Creativity Techniques Lead to Good Quality Requirements?

Following the workshop, all CCI's which had sent representatives to the workshop were asked to rate the requirements generated using the Volere measures of customer satisfaction and dissatisfaction [9]. In other words, CCIs were asked to rate, on a scale of 1 – 5, how satisfied they would be if a requirement was met in the final system (where 5 is most satisfied), and also on a scale of 1 – 5, how dissatisfied they would be if the requirement were not met (where 5 is most dissatisfied).

Data from this exercise is collated in Table 5. The table shows the numbers of times an CCI rated a requirement generated from the creativity technique or trigger shown at levels 1 – 5 for satisfaction and dissatisfaction. Since each CCI was asked to give two different ratings to each of around 200 requirements, it is perhaps not surprising that some of the requirements were not rated by some participants. In our table, we simply count and average the ratings given.

The overall averages for both satisfaction and dissatisfaction are greater than 3, suggesting that requirements generated during the creativity workshop are seen by the participants to be important in relation to the future system.

It is interesting to note that for both satisfaction and dissatisfaction, the highest average rating is for requirements generated during the round robin pros and cons session held at the beginning of the workshop. This is perhaps not surprising, as people came prepared to share their 'big ideas' about the future system, and did so during that session. So, although this session could be seen as less productive than some according to the measure shown in Table 4, it delivered, on average, the most highly rated requirements.

Table 5. Total numbers of CCI ratings of a requirement from the creativity technique or trigger shown at the level of satisfaction or dissatisfaction shown

Technique/ Creativity trigger	Customer satisfaction						Customer Dissatisfaction					
	1	2	3	4	5	Avg.	1	2	3	4	5	Avg.
Round robin pros & cons	18	14	67	50	133	3.94	12	15	54	49	94	3.88
Service	27	28	52	64	130	3.80	14	29	56	51	84	3.69
Information	30	27	54	35	49	3.23	31	22	59	30	22	2.93
Participation												
Connections	3	2	6	2	16	3.89	1	2	5	5	9	3.86
Trust	11	5	22	14	27	3.51	9	9	16	10	18	3.30
Convenience	55	32	68	32	88	3.24	46	28	61	27	62	3.13
Total	144	108	269	197	443	3.60	113	105	251	172	289	3.47

Considering the different creativity triggers used in idea generation sessions, it is also interesting to note that the triggers which were apparently most productive did not necessarily produce the most important requirements. For example, the ‘Information’ trigger was the second most productive (see Table 4), but requirements generated using that trigger had the lowest average satisfaction and dissatisfaction ratings of those from any trigger. ‘Connections’, on the other hand, appeared to be the least productive trigger according to Table 4, but to stimulate the requirements with the highest satisfaction and dissatisfaction ratings of any trigger. Once again these findings may be quite specific to this group of participants working in this domain. It is the impression of the facilitator that the ‘Connections’ and ‘Trust’ triggers were interpreted in quite a technical way (as relating to networking and security, for example), and were in this sense outside of the expertise of most of the stakeholders present. This may have accounted both for the apparently low productivity (i.e. the low number of requirements generated) and the high importance attached to the requirements generated. Further research is needed before we can generalize about the effectiveness of different creativity triggers and techniques.

5.4 Is There Any Association between the Creativity Technique or Trigger Used and the Part of the System for Which Requirements Are Derived?

There is a wide variation in the numbers of requirements identified for the different use cases, from 0 to 30, as shown in Table 6. The main foci for attention were the editing of forms on the provider side (‘Edit forms’), and making applications to attend training courses on the customer side (‘Make application’). 19 requirements were also identified in relation to customer ‘Comfort functions’ – features of the system which would make it easier and more pleasant for customers to use – and 24 requirements were identified in relation to interfaces, import/export functionalities and xml formats for linking with external systems (‘Import/export’).

Requirements from the round robin pros and cons session are particularly focused on the ‘Edit forms’ use case (which accounted for 15 out of the 41 requirements

Table 6. Association of requirements from different sources with use cases or system-level aspects of functionality

Technique/ Creativity trigger	Customer use cases							Provider use cases					System-level requirements			
	Login	Search	Display results	Make application	Payment	Offers	Comfort functions	Edit forms	Edit app. forms	Admin/monitoring	Marketing	Statistics	Performance	Usability	Import/export	Security
Round robin pros & cons		2		4			5	15	1		1			2	11	
Service		2	1	7	1	2	3	7			2	1			7	
Information			9	3		5	4		1		2	1				
Participation																
Connections											1			1	2	
Trust			1	1			2							1		4
Convenience		4	1	6		3	5	8			5				4	
Total	0	8	12	21	1	10	19	30	2	0	11	2	0	4	24	4

generated during this session) and ‘Import/export’ connections with external systems (11 out of 41). This reflects the areas of concern which the participants brought to the workshop. However, it is noticeable that requirements related to other areas of functionality were identified later in the workshop, during idea generation sessions using the creativity triggers. For example, while no requirements for the ‘Display results’ use case were identified during the pros and cons session, a total of 12 had been identified by the end of the idea generation sessions using creativity triggers. No requirements were identified in relation to ‘Offers’ in the pros and cons session, but creativity triggers lead to 10 new requirements in this area, and finally only 1 requirement relating to ‘Marketing’ was raised during the pros and cons session, but 10 new requirements were added during idea generation. This suggests that the work with creativity triggers in general gave participants the opportunity to consider broader issues and other parts of the system than those on which they might initially have focused.

Considering the impact of work with particular triggers, the spread of requirements identified using the Service and Convenience triggers appears to reflect the trends from the workshop as a whole, with most requirements from these triggers relating to ‘Make applications’, ‘Edit forms’ and ‘Import/export’. Requirements generated using other triggers do not always follow the same pattern. For example, perhaps not surprisingly, the biggest group of requirements from the ‘Information’ trigger relate to

the use case about displaying results of searches for course information. Too few requirements were identified using the Connections, Trust and Participation triggers to be able to identify any trends of this kind.

6 Lessons Learnt

This was the first use of a creativity workshop within CNM, and the first time that such a workshop had been facilitated by someone outside of the RESCUE team that originally developed the concept. The workshop proved to be an extremely useful technique in this context. Many important requirements were generated in a short space of time. In CNM's experience of similar projects, it could take around a year of monthly visits, meetings and discussions to collect a number of requirements similar to that collected through the use of the one day creativity workshop in the EDA project. While some of the efficiency gains may have come simply from collecting a number of different stakeholders together in a single workshop rather than carrying out separate meetings with the different stakeholder organisations, it is the impression of the facilitator that other benefits were due to the use of creativity techniques within the workshop. These techniques surfaced a wider range of ideas, from more different stakeholders, and generated different kinds of ideas from those which would typically be identified through the use of 'standard' requirements techniques. The feedback from participants about both their experience of using creativity techniques during the workshop and the quality of the resulting use cases and requirements was also very positive.

Based on this experience, CNM will use creativity workshops again to collect requirements for projects similar to the EDA project, where there is a need for a user-oriented requirements process to define requirements for a sizeable product or application, where requirements are initially unclear and there are heterogeneous user groups with different requirements and backgrounds (technical, organisational, content).

One important lesson concerned the management of stakeholder expectations about the requirements activities in and around a workshop. People were surprised and even resentful in the beginning as their expectations differed completely from what actually happened. They expected a meeting where they could place some ideas or just follow a presentation and then start a discussion – the way they usually define applications. Some participants initially criticised the definition of system boundaries as "useless" or a "waste of time". This was the most important, and most difficult part of the workshop. As the event progressed, the participants' understanding of why boundaries and the identification of actors are important developed. The most important lesson is to ensure that a good explanation is given as to why this kind of work is important. In future use of creativity workshops, especially with non-technical target groups who have little or no experience of the requirements process, there is a need for some easy to understand arguments and explanations of, for example, why system boundaries are important, and how actors or functionality groups will have influence.

Another lesson, based on our experience, is the need to incorporate some modifications of the creativity process in the case of projects with a clearly fixed budget limit, in order to reduce or prevent dissatisfaction. A workshop can generate many ideas, but there may not be the budget to realise them. In such cases creativity should, if possible, be channelled to focus on areas of functionality within the range of the

project budget. One possibility would be to identify extra costs in parallel with requirements so that the customer can decide whether s/he wants the relevant features or not. In the case of product development, the normal practice of CNM is to work first with a pilot customer, before developing a product for general release. In this case, the pilot customer would have the opportunity to be creative, but CNM would ultimately decide whether a particular feature should be “in” or “out of scope”.

In more general terms, the results from this workshop suggest that the effectiveness of different creativity triggers may depend on the project context, and especially on the interests and experience of the stakeholders and the nature of the system to be developed. In the workshop reported in this paper, some triggers were apparently more productive than others, in terms of the numbers of requirements generated by people working with them. Some triggers seem also to have led to the generation of more important requirements than others. However, it is important to note that the triggers which stimulated the generation of the highest numbers of requirements were not the same as those which led to the requirements which were most valued by stakeholders. Finally, there is some evidence that the use of creativity triggers during a workshop can stimulate stakeholders to identify requirements for parts of a new system on which they had not previously focused, and that some triggers (such as ‘Information’) may focus attention on particular aspects of the system. We look forward to building on these findings in future workshops.

Acknowledgements. The work reported in this paper began as part of the APOSDLE project, which is partially funded under the FP6 of the European Community within the IST work programme (project number IST-027023).

References

1. Maiden, N.A.M., Robertson, S.: Integrating Creativity into Requirements Processes: Experiences with an Air Traffic Management System. In: Proceedings of the 13th IEEE International Requirements Engineering Conference (RE 2005), IEEE CS Press, Los Alamitos (2005)
2. Maiden, N.A.M., Ncube, C., Robertson, S.: Can Requirements be Creative? Experiences with an Enhanced Air Space Management System. In: Proceedings of the 29th International Conference on Software Engineering (ICSE 2007), IEEE CS Press, Los Alamitos (2007)
3. The APOSDLE project web site (2007), <http://www.aposdle.org>
4. Ley, T., Kump, B., Lindstaedt, S.N., Albert, D., Maiden, N.A.M., Jones, S.: Competence and Performance in Requirements Engineering: Bringing Learning to the Workplace. In: Proceedings of the 2nd Workshop on Learner-Oriented Knowledge Management & KM-Oriented E-Learning (LOKMOL) (2006)
5. Jones, S., Maiden, N.A.M.: RESCUE: An Integrated Method for Specifying Requirements for Complex Socio-Technical Systems. In: Mate, J.L., Silva, A. (eds.) Requirements Engineering for Sociotechnical Systems, Idea Group Inc. (2005)
6. Jones, S., Maiden, N.A.M., Karlsen, K.: Creativity in the Specification of Large-Scale Socio-Technical Systems. In: Golightly, D., Rose, T., Wong, B.L.W., Light, A. (eds.) Proceedings of CREATE 2007, the Conference on Creative Inventions, Innovations and Everyday Designs in HCI, London, UK, June 13-14, 2007, pp. 41–46. Ergonomics Society (2007)

7. IHK24 Partner homepage (2007), <http://www.ihk24.de>
8. Andrews, D.C.: JAD: A Crucial Dimension for Rapid Applications Development. *Journal of Systems Management*, 23–31 (March 1991)
9. Robertson, S., Robertson, J.: *Mastering the Requirements Process*. ACM Press, New York (1999)
10. Maiden, N.A.M., Robertson, J.: Creative Requirements - Invention and its Role in Requirements Engineering, Tutorial Notes. In: RE 2005 Conference, Paris, France (August 29, 2005)

Can We Beat the Complexity of Very Large-Scale Requirements Engineering?

Björn Regnell^{1,2}, Richard Berntsson Svensson², and Krzysztof Wnuk²

¹ Sony Ericsson, Lund, Sweden

<http://www.sonyericsson.com>

² Lund University, Sweden

bjorn.regnell@cs.lth.se,

<http://www.cs.lth.se>

Abstract. Competitive development of complex embedded systems such as mobile phones requires management of massive amounts of complex requirements. This paper defines and discusses orders of magnitudes in RE and implications of the highest order of magnitude that we have experienced in industrial settings. Based on experiences from the mobile phone domain we propose research areas that, if addressed successfully, may help beating the complexity of Very Large-Scale Requirements Engineering.

1 Introduction

The complexity and size of software-intensive systems continues to increase, which in turn gives increasingly large and complex sets of requirements. How many requirements can an industrial system development organisation manage with available Requirements Engineering (RE) processes, methodology, techniques and tools? This is hard to know as RE research often falls short in characterizing the scalability of proposed methods. How large and complex sets of requirements do we need to consider when researching new RE technology? We have no complete picture of current industrial practice in terms of complexity of sets of requirements, but we have experiences from industrial cases with enormous complexity where current RE technology have useful but partial effect [4,5,6]. Our objective with this paper is to share some important research opportunities that we have found in our observation of what we call Very Large-Scale Requirements Engineering (VLSRE).

The paper is organized as follows. Section 2 proposes a definition of VLSRE based on the size of a requirement set as a proxy for its complexity. Section 3 provides a case description of the mobile phone domain that illustrates an instance of VLSRE. Section 4 highlights some research opportunities that we through our own industrial experience have found relevant to VLSRE. Section 5 concludes the paper.

2 Orders of Magnitude in Requirements Engineering

Table 1 defines four orders of magnitude in RE based on the size of the set of requirements that are managed by an organisation that develops software-intensive

Table 1. Three orders of magnitude in Requirements Engineering

<i>Abrev.</i>	<i>Level</i>	<i>Order of magnitude</i>	<i>Sample empirical evidence</i>	<i>Interdependency management conjectures with current RE technology</i>
SSRE	Small-Scale Requirements Engineering	~10 requirements		Managing a complete set of interdependencies requires small effort.
MSRE	Medium-Scale Requirements Engineering	~ 100 requirements	[3]	Managing a complete set of interdependencies is feasible but requires large effort.
LSRE	Large-Scale Requirements Engineering	~1000 requirements	[8]	Managing a complete set of interdependencies is practically unfeasible, but feasible among small bundles of requirements.
VLSRE	Very Large-Scale Requirements Engineering	~10000 requirements	[6]	Managing a complete set of interdependencies among small bundles of requirements is unfeasible in practice.

systems. The levels are inspired by the characterisation of orders of magnitude in integration of digital circuits.

We have chosen *numbers of requirements* as a proxy for complexity as we believe that increased numbers of customers, end users, developers, subcontractors, product features, external system interfaces, etc. come along with increased number of requirements generated in the RE process as well as increased complexity of RE. Furthermore, in almost all industrial RE settings that we have encountered, the requirements that are documented are also eventually enumerated and often given a unique identity, allowing a counting of the elements in the set of requirements in a given development organisation. If so, it is fairly easy to give a size figure for a given case that in turn allows for cases to be compared in terms of their order of magnitude (although the average level of detail in the set of requirements needs to be fairly similar for the comparison not to be too speculative).

We suggest based on experience that the complexity of a set of requirement is heavily related to the nature of interdependencies among requirements (see e.g. [2] for an empirical investigation of interdependencies). With a realistic degree of interdependencies among n -tuples of requirements, we hypothesize that the number of interdependencies to elicit, document and validate increases dramatically with increased number of requirements. When shifting from MSRE to LSRE, a typical heuristic for dealing with the complexity of interdependency management is to bundle requirements into partitions and thereby creating a higher level of abstraction where interdependencies among bundles can be managed with reasonable effort. When shifting from LSRE to VLSRE, our conjecture is that even the number of bundles gets too high and the size of bundles becomes too large to allow for interdependency management with desired effectiveness. If the requirements bundles become too large, the interdependency links loose practical usefulness as they relate too coarse grained abstractions.

SSRE and MSRE is a common scale in research papers that seek to validate a proposed method or tool. For example, in [3] the scalability issue is addressed but for a specific tool dealing with only 67 requirements. In this situation it is possible to

enumerate and manage complex relations among requirements even with dense relation patterns. However, we believe that few industrial situations in current system development can avoid stretching beyond SSRE and even MSRE. We have found few examples in RE literature that discusses LSRE (such as [8]), but we believe that LSRE is common industrial practice (confirmed by [1]). We also believe that a significant number of companies that currently face LSRE will grow into the situation of VLSRE as their products grow in complexity, their product portfolio grows in size, and they introduce product line engineering that further drives RE complexity. In the next section we describe one specific case that already has experienced such a transition.

3 A Case of VLSRE

To illustrate the complexity in VLSRE we provide a case description of embedded systems engineering in the mobile phone domain, based on experiences at Sony Ericsson, which has faced a transition from LSRE to VLSRE in the last years, while remaining competitive on the market with a growing number of around 6000 employees. Mobile phones include a wide range of features related to e.g. communication, business applications and entertainment. The technological content is complex and includes advanced system engineering areas such as radio technology, memory technology, software design, communication protocols, security, audio & video, digital rights management, gaming, positioning etc. The complexity of RE is driven by a large and diverse set of stakeholders, both external to the company and internal. Table 2 gives examples of stakeholders that generate requirements.

Table 2. Examples of stakeholders that generate requirements

<i>External Stakeholders</i>	<i>Internal Stakeholders</i>
Competitors	Accessories
Consumers of different segments	Customer Services
Content providers	Market research
Legislation authorities	Marketing & customer relations
Operators	Platform development (SW+HW)
Retailers	Product, application & content planning
Service providers	Product development (SW+HW)
Share holders	Product management
Standardization bodies	Sourcing, supply & manufacturing
Subcontractors & component providers	Technology research & development
	Usability engineering

Some stakeholders are counted in billions, such as consumers of different segments, while some are counted in hundreds such as operators. In the case of Sony Ericsson, the requirements that are generated from internal and external stakeholders amount to several tens of thousands, and this is a clear case of VLSRE.

Figure 1 provides a simplified picture of the different types of requirements and their relations. Similar to the case in [3], requirements originating from external stakeholders (called *market requirements*) are separated from but linked to *system requirements* that are input to platform scoping in a product line setting. Market

requirements are mainly generated by operators submitting specifications with thousands of requirements that require statements of compliance. The total volume of market requirements at Sony Ericsson exceeds 10000 as well as the total volume of platform system requirements. In order to make scoping feasible, platform system requirements are bundled into hundreds of *features* that represent the smallest units that can be scoped in or out. In order to support product development the platform capabilities are organised into *configuration packages* that improve over time as more and more features are implemented for each new version of a platform. Products are configured through assemblies of configuration packages according to the rules of how they can be combined based on their interdependencies. All categories of requirements are expressed in natural language text and include a set of attributes according to a requirements data model for a requirements data base implemented in a commercial requirements engineering tool. Based on our experience with the complexity of this VLSRE case we bring forward three key research opportunities in the next section.

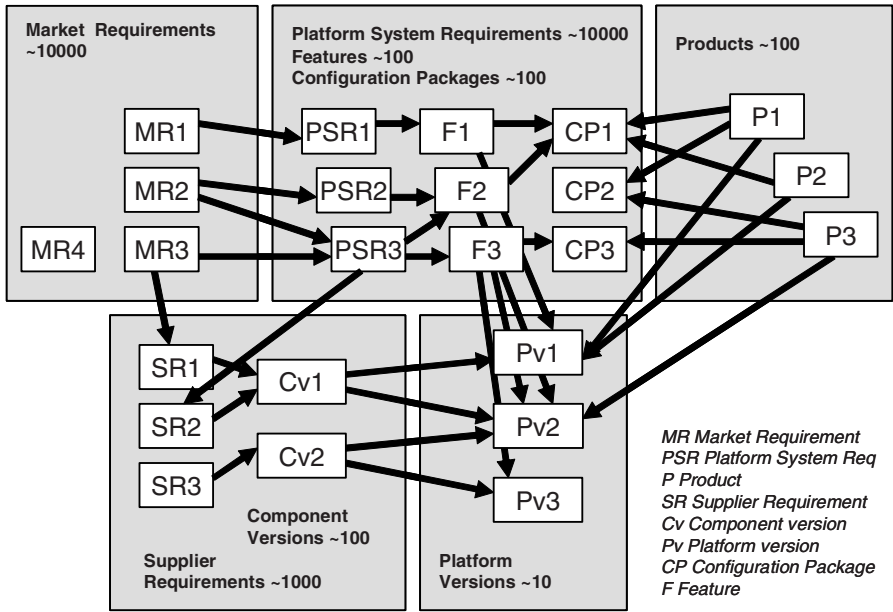


Fig. 1. Orders of magnitude in different artifacts of a specific VLSRE case

4 Three Key Research Opportunities in VLSRE

Based on our experience from working several years in the previously described VLSRE context, we have chosen to highlight three areas where we believe RE research can and should contribute:

- **Sustainable requirements architectures: Fighting information overload.** With the term requirements architecture we mean the underlying structure by which the

requirements are organised including the data model of the requirements with their pre-conceived attributes and relations. In VLSRE, the amount of information that must be managed is immense and not possible to grasp in all its details by a single person. In order to fight information overload we need requirements architectures that are sustainable in the sense that they allow for controlled growth while allowing the requirements engineers in a large organisation to keep track of the myriad of issues that continuously emerge. How should we design sustainable requirements architectures? Which concepts are stable? Which attributes and links are most important to maintain? What is the simplest yet competitive requirements data model?

- ***Effective requirements abstractions: Fighting combinatorial explosions.*** In VLSRE situations where interdependencies among requirements are critical (such as prioritisation, resource estimation, and change impact analysis) we inevitably stumble on combinatorial explosions, further fuelled by product line engineering that significantly increases the complexity of the requirements architecture. Finding all interdependencies among 20 requirements is possible, but not among 10000. A major vehicle for fighting this is abstraction mechanisms and experience-based heuristics. In interviews with requirements architects at Sony Ericsson we encounter heuristics related to requirements bundling and choice of level of detail, but they still often struggle to find yet another needle in the haystack. Can we empirically characterize the effectiveness of requirements abstractions? How can we empirically investigate human requirements comprehension? How to support humans in navigating and searching massive sets of requirements? How can we make relevant visualisations of different partial viewpoints on immense requirements heaps that hide irrelevant details but highlight important issues for a given decision-making situation? What level of uncertainty and degree of approximation can we tolerate?
- ***Emergent quality predictions: Fighting over-scoping.*** Given a competitive market and a large and demanding set of stakeholders, there seems to be an inevitable shortage of resources to meet quality expectations. To predict the system level quality aspects that emerge from a myriad of details is very difficult and we have seen a sustained risk of defining a too large scope for platform development partly due to the inherent difficulty in understanding quality requirements and predicting their impact and required development resources. We are beginning to understand how to do roadmapping and cost-benefit analysis of quality requirements in sub-domains [7], but we still struggle with how to manage a holistic view where quality requirements are aggregated to system level. How can we deal with interdependencies among quality requirements? Maybe we can get the scope of functions right, but are the set of functions of adequate quality? How can we with reasonable effort prioritize emergent system qualities when predictions are uncertain?

5 Conclusion

During the last decade we have seen VLSRE emerge as a very demanding challenge. Parts of the embedded systems engineering industry are facing severe problems in coping with the rapidly increasing complexity of the massive amount of information that needs to be managed in order to be competitive on the market. Our conjecture is that we have hit the roof with current tools and we need to mobilise RE researchers to try to beat the

complexity of VLSRE. We should also increase our knowledge of how existing methods, tools and techniques perform in SSRE, MSRE, LSRE and VLSRE respectively, to better understand which methods that are good candidates for use in VLSRE combined with sustainable requirements architectures and effective requirements abstractions. By advancing these techniques and heuristics we might be able to manage the complex task of predicting emergent system quality aspects already in the early stages of the development cycles where opportunities are rising while uncertainties are high.

Acknowledgements. This work is supported by VINNOVA (Swedish Agency for Innovation Systems) within the MARS and UPITER projects. Special thanks to Thomas Olsson for input on numbers and to Even-André Karlsson for input to the visualisation of entity relations in fig. 1.

References

1. Brinkkemper, S.: Requirements Engineering Research the Industry Is and Is Not Waiting For. In: 10th Anniversary Int. Workshop on Requirements Engineering: Foundation for Software Quality. Riga Latvia, pp. 41–54 (2004), URL visited 07/12/2007 <http://www.sse.uni-essen.de/refsq/downloads/refsq-10-booklet.pdf>
2. Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., Nattoch Dag, J.: An industrial survey of requirements interdependencies in software product release planning. In: Proc. IEEE Int. Conf. on Requirements Engineering, Toronto, Canada, pp. 84–91 (2001)
3. Feather, M.S., Cornford, S.L., Gibbel, M.: Scalable mechanisms for requirements interaction management. In: Proc. 4th Int. Conf. on Requirements Engineering, Los Alamitos, USA, pp. 119–129 (2000)
4. Natt och Dag, J., Gervasi, V., Brinkkemper, S., Regnell, B.: Speeding up Requirements Management in a Product Software Company: Linking Customer Wishes to Product Requirements through Linguistic Engineering. In: Proc. of the 12th Int. Requirements Engineering Conf., Kyoto, Japan, pp. 283–294 (2004)
5. Natt och Dag, J., Gervasi, V., Brinkkemper, S., Regnell, B.: A Linguistic Engineering Approach to Large-Scale Requirements Management. IEEE Software 22, 32–39 (2005)
6. Regnell, B., Olsson, H.O., Mossberg, S.: Assessing requirements compliance scenarios in system platform subcontracting. In: Proc. 7th Int. Conf. on Product Focused Software Process Improvement, Amsterdam, The Netherlands, pp. 362–376 (2006)
7. Regnell, B., Höst, M., Berntsson Svensson, R.: A Quality Performance Model for Cost-Benefit Analysis of Non-Functional Requirements Applied to the Mobile Handset Domain. In: Sawyer, P., Paech, B., Heymans, P. (eds.) REFSQ 2007. LNCS, vol. 4542, pp. 277–291. Springer, Heidelberg (2007)
8. Park, S., Nang, J.: Requirements management in large software system development. In: IEEE Conf. on Systems, Man, and Cybernetics, New York, USA, pp. 2681–2685 (1998)

Macro-level Traceability Via Media Transformations

Orlena C. Z. Gotel¹ and Stephen J. Morris²

¹ Department of Computer Science, Pace University, New York, USA
ogotel@pace.edu

² Department of Computing, City University, London, UK
sjm@soi.city.ac.uk

Abstract. This paper proposes an alternative approach to the examination of artifacts whose contents must be traceable to promote software quality. The approach places emphasis on media use and media transformations. We suggest that one cannot begin to assign and sustain traceability relations at a micro-level between units of content if the sign systems that have been created and transformed to represent this content are not considered at a more macro-level. Our approach is anticipatory, feasible to automate and exemplified.

Keywords: Media Transformation, Multimedia, Requirements Traceability.

1 Introduction

Recent traceability research has focused on establishing links between semantically similar terms to identify automatically content-based traceability relations between the artifacts of software development [2]. These approaches account for textual artifacts and, to a limited extent, the textual characteristics of structured diagrams. While they address some of the problems associated with traceability [5], they ultimately lend themselves to natural language ambiguity and many artifact types are precluded. The premise of our work is that artifacts relevant to the trace record will be held in multiple media in the future, especially those generated during upstream requirements-related activities, including the results of observational studies or sketches drawn by stakeholders. Video fragments from elicitation sessions are already used to provide supporting rationale for requirements in some contexts [3] and a vision of video-based requirements engineering continues to gain clarity [7].

Presuming a media-rich software development environment, we suggest that you first have to be precise about the nature of the relation between the underlying media types before you can say what the implications are for content change and any particular traceability relation between artifacts. The underlying assumption is that there is no such thing as a pure element of content, only some representation of it as an artifact. It is therefore essential to understand the process whereby representations come into being and are transformed because this is the only way to understand what happens to content. We extend previous research by marrying traceability with multimedia production to propose an approach through which to make decisions about media choices, combinations and transformations when seeking to create or recover a representative trace record [6]. The concept for the approach is exemplified to highlight its potential value in framing a familiar topic from a new perspective.

2 Media Use in Requirements Engineering

Any subject matter being communicated has an associated medium which is its carrier, or vector, in the physical sense. In a second sense media are abstract; they are agencies for the communication of subject matter. As such they are separate sign systems, the most common and most significant being natural language text and speech, graphics, still and moving images, and specialised systems such as numbers, mathematical and computer languages [9].

Fig. 1 shows how media may be involved in a simple requirements engineering scenario. During an elicitation interview, the respondent can indicate other relevant people, activities or documents. The range of potential responses and their referents suggests the need for text, video and sound recording. A minimal set of other material (e.g. an operations manual and a client's briefing document) represents sources in conventional print media. If the interviewer asks questions from a pre-written questionnaire, the abstract medium is written natural language and the physical medium is ink on paper. The spoken questions and answers comprise spoken natural language carried by sound waves. If the interview is recorded, moving pictures, natural language speech and sounds proper to the domain (e.g. traffic) are captured. Responses and supplementary detail may be recorded as written natural language text and images on paper. The primary source material is thus rich in media.

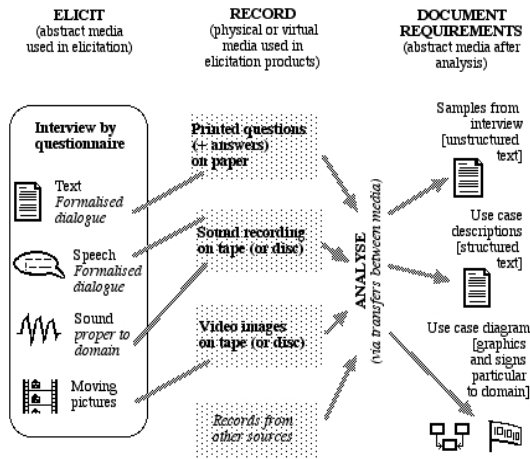


Fig. 1. The media in a simple requirements engineering scenario

During what is loosely defined as ‘analysis’, a series of media transfers take place. An audio segment may be transcribed and augmented with contextual information from the video to gain gesture and gaze detail. Progressively, the abstract media will be reduced in variety and from amongst the abstract media initially involved only text survives when requirements are first formally documented. Moreover, only the textual samples from the original interview can be unequivocally traced back to an original source. Although every output is now within the single digital medium, the complexities and implications of media transfers remain hidden.

3 Media Transformations

In earlier work, the concept of media transformations for describing and prescribing changes to abstract media elements was developed [8]. We give examples below as they apply to the scenario above and explain the implication for traceability.

Origination: Examples would be text of the questionnaire responses, recorded video images, speech and other sounds from the interview environment, text and graphics in the operations manual, or text and images in the brief. This transformation should guarantee the identification of all primary source material and provide starting points for forwards traceability and all end-points for backwards traceability.

Amplification: An example would be the elaboration of the interview text transcribed from the audio recording with the text notes of the visual indicators in the video. Use should provide the opportunity to identify partial content changes, not involving the complete merger or amalgamation of media that might otherwise be without an identifiable source, along with the ability to retrace provenance in the original context. This may be important for forwards and backwards traceability where clarity of change is significant and wider history is relevant to understanding.

Revision: Examples would be alterations to the text of the questionnaire responses to ensure they reproduce the recorded speech, or the structuring of pre-existing text to form use case descriptions. Differentiating from amplification, elements of an artifact are completely replaced by elements in the same medium as opposed to extended. This transformation demands identification of the basis for any revision, even if this requires the origination of a primary source. The implication is the possible need to incorporate an element of rationale into the trace record.

Translation: Examples would be the translation of interview speech to written text, the video images to text of the content, or the use case descriptions to the use case diagram. Switching between abstract media involves representation in an alternative sign system and there are implicit losses involved [1]. This is problematic, and irreversible for backwards traceability, where signs are wholly undifferentiable one from another either syntactically or semantically [4]. The translation from the video of the interview into text would be subject to such restriction. Even recorded speech to text, although guided by transcription conventions, comes with some losses.

Outline: An example would be the list of use cases from the text in the operations manual. With neither abstract medium nor domain of content changing, this transformation should not cause problems for traceability, but is subject to the restriction that detail is lost. Where the media vary and are reduced in number, say from a video, speech and sound recording to text, it is important to know for traceability whether the outline represents indexical properties.

Merger: An example would be text from the interview answers combined with text derived from extraneous dialogue. This transformation indicates the fusion of pre-existing paths, henceforth treated as a single path. The complete merger of elements in the same abstract medium should not be problematic unless it is important to differentiate between the contributions of sources, to propagate forward changes from things prior. Traceability implications depend on the volatility of contributing paths.

Amalgamation: An example would be the fusion of some of the text from the use case description with graphics to form the use case diagram. The process of specification is one in which the number of abstract media used is steadily reduced,

often to text and notational sign systems such as UML. Only if the separate abstract media remain differentiable is the traceability in either direction unimpaired.

Proxy creation/use: An example of creation would be the questionnaire text to be spoken; use would be the questions spoken. Tracing depends on an accurate mapping between elements in the different abstract media (e.g. direct mappings between text and the spoken version). Proxy transformations clarify the role of artifacts, such as a storyboard standing in place of a design, but present issues for any trace.

Substitute creation/use: An example of creation would be any UML representation that is an intermediate artifact between requirements and implementation; use would be transitional UML models prior to implementation. These transformations play an important role in the exploration of concepts and promote trace continuity.

Compare use: When one artifact influences another, but does not participate in a media transformation, it acts as a basis for comparison. This transformation contributes to the accuracy of others, thus to the completeness of the trace record, and so to the effectiveness of traceability. An example would be the recorded speech used for comparison with the written answers to the questionnaire to check for accuracy.

4 Media-Based Traceability

In software engineering, improving lifecycle-wide traceability involves defining an exact path for tracing and understanding the alteration of information content along this path over time. Where this involves a range of abstract media representations, carried by physical and digital media, there are many paths that might be followed. One set of paths for the scenario is given in Fig. 2. It shows how media transformations may provide a practical means of linking artifacts at a high level.

The media transformations in the scenario begin with origination, shown by the creation of primary sources via the interview or preparatory tasks (1.1-1.5). Given the varied abstract media, translation into a common abstract medium may be necessary to enable the construction of new artifacts. Switching between abstract media is a translation (2.1, 2.2) and involves content discontinuity. The trace path and record will be impaired where translation is undertaken, implying the potential need to revisit original sources. Whether an artifact is being used as a subsequent compare in its original medium or in a translated form also has implications for what is re-examinable. Translation presents a problem for traceability since it is not bi-directional without effort. When there are choices between using media other than natural language text, we should be able to determine which media are going to be the most problematic and know what measures to take to preserve future traceability.

Four transformations affect changes within a single abstract medium: amplification, revision, outline and merger. The scenario shows an amplification (2.3) where an element in one medium elaborates another of the same medium and a revision (2.4 etc.) where an element replaces one of the same medium. One would expect to see many such transformations within software development since its essence is the distillation of content. Both these transformations demand versioning information for understanding. Amplification typically requires a subsidiary artifact to ensure backwards traceability. This is not so for revision, which may simply follow from changes to the prior artifact. However, if the basis for the revision is relegated to memory, rationale is not automatically retrievable and the path not easily reversible.

1 Primary source material [in multiple media and multimedia]
0.1 proxy creation: text of questionnaire to be spoken in interview
0.2 proxy use: questions spoken in interview
1.1 origination: text of responses to questionnaire
1.2 origination: recorded video images
1.3 origination: speech and other sounds from interview and work environment
1.4 origination: text and graphics in operations manual
1.5 origination: text and images in client brief
2 Conversion to text [transferring back to the 'primary modelling language' of natural language] – now within digital carrier media
2.1 translation: speech in recorded interview 1.3 to written text
2.2 translation: video images 1.2 to text notes of content
2.3 amplification: elaboration of interview text 1.1 with <i>compare use</i> of text notes of video 2.2
2.4 revision: alterations to text of responses 2.3 with <i>compare use</i> of speech recorded 1.3
2.5 merger: text from structured interview answers with text derived from any open-ended exchanges or extraneous dialogue 2.1 and 2.4
2.6 outline: text list of possible use cases from interview text 2.5
2.7 outline: text list of possible use cases from operations manual 1.4
2.8 outline: text list of possible use cases from client brief 1.5
3 Documentation [transferring to a 'secondary modelling language' of structured natural language]
3.1 revision: structuring of interview text 2.5 to form initial use case descriptions
3.2 revision: structuring of operations manual text 1.4 (and possible <i>translation</i> from graphics) to form initial use case descriptions
3.3 revision: structuring of client brief text 1.5 (and possible <i>translation</i> from images) to form initial use case descriptions
3.4 merger and revision 2.6, 2.7, 2.8: use case list
3.5 merger and revision 3.1, 3.2, 3.3: use case descriptions
3.6 proxy creation: use case descriptions 3.5 in part for later UML activity diagrams
3.7 revision: sample extractions from interview text 2.5 on basis of <i>compare use</i> of use case list 3.4
4 Modelling [transferring to a 'tertiary modelling language' using non-textual components as the foundation for representation]
4.1 outline and translation: use case list 3.4 to use case names and diagram elements; use case descriptions 3.5 to associations in diagram
4.2 amalgamation and substitute creation: elements from 4.1 brought together in a use case diagram (or model)

Fig. 2. Traceability from a media-based and transformational perspective

The outline transformation, used in the scenario to structure use case descriptions (2.6-2.8), provides a précis version without media change. This appears innocuous, but if the source artifact combines abstract media, the textual outline may be a result of implicit translations coupled with additional undocumented information derived from the juxtaposition of media. An outline in one medium derived from multiple media presents a potential break in continuity for the trace record and may need to be re-examinable in its wider derivation context.

When two or more elements in the same medium are combined to form another, the result is a merger. While not directly reversible, merger transformations within a single abstract medium minimize content loss. The first merger (2.5) combines text that has been translated from speech with text elaborated with textual information derived from video. This artifact could equally have been constructed as a result of revisions/amplifications, with *compare use* of primary sources, but this would have had negative consequences for the trace record because the absence of discrete intermediary stages compounds impact analysis. Where a merger takes place with an accompanying revision/amplification (3.4, 3.5), the traceability path is likely to be jeopardized unless a supporting artifact is provided.

Amalgamation transformations (4.2) should be common in requirements engineering since they are used when constructing use case models. However, the combination of media elements of different types, whatever the medium of the result, may be even more unpredictable in outcome than translation. This is the case where it is difficult to untangle the contributing media elements and their individual paths for traceability. The choices made as to the types and ordering of transformations is of interest because amalgamation can be avoided, for example, by translation into a common medium (e.g. text in the scenario) followed by merger. Such ordering will alter the path through which requirements are engineered, in one case retaining some ability to retrace separate contributing paths. Amalgamations are worth examining if preserving the integrity of the trace record is crucial and effort should be made to retain separation potential if parts of the embedded content are likely to change.

5 Towards a Framework for Macro-level Traceability

There are implications for content loss or gain when different media types are used in software development, impacting traceability. Decisions need to be made as to the use of media, combinations of media and the ordering of transformations between media as content from those artifacts constituting primary source material is created and transformed into specification and code. Our research seeks to provide a framework and develop guidelines to help engineers take these decisions in support of their anticipated traceability needs. We suggest that an understanding of the artifact collection, at a foundational and representational level, is critical for contextualizing more discerning forms of traceability, irrespective of manual provision or automated recovery. The routine generation of macro-level traceability between media-rich artifacts should not be an insurmountable task for future requirements management environments and integrated guidelines would alert to and help mitigate critical traceability issues, focusing effort only when and where most needed given the potential costs incurred. An exploration of these important topics, accompanied by validation of the underlying approach, forms our on-going research.

References

1. Anderson, P.B.: *A Theory of Computer Semiotics*. 2nd edn. CUP, Cambridge (1997)
2. Cleland-Huang, J., Settini, R., Romanova, E., Berenbach, B., Clark, S.: Best Practices for Automated Traceability. *IEEE Computer* 40(6), 27–35 (2007)
3. Gall, M., Bruegge, B., Berenbach, B.: Towards a Framework for Real Time Requirements Elicitation. In: 1st Intl. Workshop on Multimedia Requirements Engineering (with 14th IEEE Intl. Requirements Engineering Conference), Minneapolis, MN (2006)
4. Goodman, N.: *Languages of art: An approach to a theory of symbols*, 2nd edn. Hackett, Indianapolis, IN (1976)
5. Gotel, O.C.Z., Finkelstein, A.C.W.: An Analysis of the Requirements Traceability Problem. In: 1st IEEE Intl. Conference on Requirements Engineering, Colorado Springs, CO, pp. 94–101 (1994)
6. Gotel, O.C.Z., Morris, S.J.: Crafting the Requirements Record with the Informed Use of Media. In: 1st Intl. Workshop on Multimedia Requirements Engineering (with 14th IEEE Intl. Requirements Engineering Conference), Minneapolis, MN (2006)
7. Jirotko, M., Luff, P.: Supporting Requirements with Video-Based Analysis. *IEEE Software* 23(3), 42–44 (2006)
8. Morris, S.J.: Media transformations for the representation and communication of multimedia production activities. In: Sutcliffe, A., et al. (eds.) *Designing Effective and Usable Multimedia Systems*, pp. 72–85. Kluwer Academic Publishers, Norwell Mass (1998)
9. Morris, S.J., Finkelstein, A.C.W.: Engineering via discourse: Content structure as an essential component for multimedia documents. *International Journal of Software Engineering and Knowledge Engineering*, World Scientific Pub. Co. 9(6), 691–724 (1999)

Towards Simulation-Based Quality Requirements Elicitation: A Position Paper

Roland Kaschek¹, Christian Kop², Vladimir A. Shekhovtsov³, and Heinrich C. Mayr²

¹ School of Engineering and Advanced Technology,
Massey University, New Zealand
Roland.Kaschek@ieee.org

² Institute for Applied Informatics,
Alpen-Adria-Universität Klagenfurt, Austria
{chris,mayr}@ifit.uni-klu.ac.at

³ Department of Computer-Aided Management Systems,
National Technical University “Kharkiv Polytechnical Institute”, Ukraine
shekv1@yahoo.com

Abstract. The future users of a system under development are not necessarily good at talking about the quality they require of that system if they cannot yet experience it. We therefore propose to support them by a simulation of the system under development thus allowing them to experience and validate system quality. Requirements are supposed to be expressed in a user-centered glossary-based semantic model.

1 Introduction

The traditional approach to requirements elicitation focuses on functional requirements. This might be justified by conceptualizing computers mainly as machines for executing operations. It is, however, unfortunate because it contributes to losing sight on alternative solutions. However, the convenience of use of a software system strongly affects its success. We therefore suggest to introduce a computer usage model in addition to the computational one. The latter is important with respect to assessing complexity of computational procedures but is not as effective with respect to requirements elicitation or specification.

Non-functional requirements (NFR) are not always clearly distinguished from functional requirements [9, 12]. We thus prefer using the term *quality requirement* (QR) instead at least for those system aspects which address system quality. We conceptualize system quality as fitness for use under stated or implied conditions [14]. One system quality aspect is the appropriateness of the provided functionality. Further quality aspects are e.g. learnability, maintainability, memorizability, performance, safety, or security. They concern the ways of use of the provided functionality. Clearly some of these ways are preferable over others.

Three arguments are in favor of the claim that specifying the “right” quality for a system under development (SUD) may be difficult. First, an assessment of an SUD’s fitness for use often is a reflection of a respective consensus among various stakeholders. Second, such an assessment depends on the anticipated or implemented

interaction of the SUD with its environment. Such an assessment obviously may be a complex task for complex SUD environments. Third, talking about suitability of SUD usage processes that one cannot experience yet is quite speculative.

An online simulation tool supporting SUD stakeholders in experiencing their (future) working environment might help in such situations: In [20] such a system, called POSE (*parameterized online simulation environment*) was introduced. POSE utilizes a specific user-centered conceptual model called QAPM (*quality-aware predesign model*) for representing the quality requirements in an easy-to-understand way. QAPM is based on KCPM, the Klagenfurt Conceptual Predesign Model, which was first introduced in [16].

This position outlines the overall methodology. Related work is shortly discussed in section 2. We then describe QAPM and POSE in sections 3 and 4, respectively, and provide a short look on further work to do in section 5.

2 Related Work and Research Issues

2.1 Traditional Quality Requirements Elicitation Techniques

Stakeholder-involving techniques of eliciting quality requirements employ traditional techniques such as interviews, brainstorming, and checklists. Usually they work by writing up requirements and structuring them based on human interaction with stakeholders. Goal-oriented techniques [4] classify the requirements according to structured system goals. Requirements Description Language (RDL) [3] represents the requirements via an XML-based model allowing for requirements composition. Quality Attribute Workshops [2] use case specific interpretations of the quality attributes for the given SUD. Glossary-based approaches [6, 16] use specific glossaries for modeling and organizing requirements.

Specification-based techniques use informal or structured requirements specifications as elicitation sources. NLP techniques can be used to elicit the quality requirements from these documents in an automated way [1, 3, 5]. The problem with these approaches is that stakeholder participation in the elicitation process is limited. In fact, after this process is completed, the stakeholders often still need to verify its results.

The first research issue arises out of an investigation of these approaches: we argue that it makes sense to give stakeholders a chance to get an experience of working with the targeted SUD prior to participating in quality requirements elicitation activities.

2.2 Using Simulations to Elicit Requirements

Early attempts to support requirements elicitation by exemplification were made in the 80ies [8, 17] under the title of (rapid) prototyping. In particular, “horizontal” prototypes (lacking implemented functionality) were introduced to simulate user interfaces in order to allow stakeholders to experience their future environment. However, due to technical limits within that time, the approaches did not leave the laboratory status. In contrast to that, today several tools (based on research projects [7, 10, 11, 19] and developed in industry [13, 18]) exist that aim at using interactive SUD *simulations* to work with system requirements.

Simulation goals. In most cases, the tools use simulation to support the requirements validation. For example, in [10, 11] an already-built requirements model is validated, in [19] such model is validated during its incremental building, in [7] the object of validation is a structure of an already-designed component and its interactions with an environment. Industry-based tools [13, 18] allow non-programmers to build and execute models simulating the external behavior of the SUD to receive feedback about the quality of the simulated interface and the required functionality as seen via this interface; they allow eliciting some subset of the quality requirements (usability, user-friendliness etc.), but only as informal user notes.

Common characteristic of these tools is that they are not specifically targeting the elicitation of quality requirements. Actually, they leave our first research issue unaddressed, as we do not know about any approach using simulation to elicit the required system qualities based on users' experience of working with this simulation. As a result, we can state *the second research issue*: we argue that it makes sense to create an environment executing simulations specifically aimed at eliciting quality requirements.

Simulation scope. These tools in most cases *simulate a SUD only as a standalone system*. Integrating these simulations into the SUD usage processes is not well supported. For example, in [10], using proposed control language, it is necessary to develop usage scenario manually for every simulation run. Describing the usage processes in this situation is similar to coding business processes in a general-purpose programming language instead of a specialized one (for example, BPEL). Two approaches are closer to addressing this problem as they pay stronger attention to the SUD usage modeling. In [7], several software process activities related to SUD (treated as a software component) are modeled (replacement, upgrading etc.). However, they are still not addressing the SUD usage processes. In [18], the user can specify the usage processes interactively using BPMN-like notation, but the goal of this specification is purely descriptive since the processes are not supposed to be simulated.

As a result, we can formulate *the third research issue*: we argue that it makes sense to implement a simulation of the entire environment for the SUD. The roles of both the user and the SUD should be completely specified in this environment.

2.3 Our Propositions

For overcoming the above problems, we propose to combine traditional techniques (focusing on documenting and discussing quality requirements) with interactive simulation (focusing on elicitation and on assessment of requirements drafts).

The next sections, therefore, introduce QAPM as such a traditional technique and discuss how it can work together with a simulation technique that addresses some of the above research issues.

3 The Quality-Aware Predesign Model QAPM

QAPM is based on the user-centered glossaries of KCPM [16] which help stakeholders to find missing information. SUD quality is modeled as a hierarchy of quality following ISO/IEC 9126-1 [15]. It takes into account the interrelationships between

these characteristics and the points of view of different stakeholders' categories. The model also reflects the fact that relationships between quality requirements and system functionality are crosscutting in nature (i.e. a single quality requirement can be related to multiple functional elements of the system). The semantics of quality and functional requirements, together with the semantics of their crosscutting relationships is collected in a structured way – as in KCPM.

Table 1 shows a part of the QAPM glossary for a quality requirement related to the response time for all actions involving orders. It is represented as a constraint referring to both the quality characteristic and the functional element of the model.

Table 1. Part of the QAPM glossary representing a quality requirement

Quality characteristic	Sequencing	Functional element	Decision operator	Threshold	Applicability	Description
Response time	WRAP	Order	<	0.5 sec	Peak hours	Users' opinion

We can build POSE on top of this semantic model. It provides (1) the definition of quality for POSE, (2) the means of representing the SUD functionality and its initial qualities, (3) the positions where the quality assessments can be made, (4) the way of expressing the semantics of elicited quality requirements.

4 The Parameterized Online Simulation Environment POSE

Actually, two kinds of simulations are supported by POSE [20]: the simulation of SUD behavior and the simulation of its usage in the particular organization.

SUD usage processes. As we consider an organization as a system that enacts a number of business processes, for simulating SUD usage we need to model the structure of the organization at hand, the resources it utilizes, as well as its business processes, supporting- and management processes. These processes are actually SUD usage processes. To save implementation effort, we propose to represent them suitably for BPM simulation engines. Therefore, POSE stores process models using the process assembler tool (PA) [21], thus allowing to use various process modeling languages (PML). We plan to have an archive of models for different industries managed by PA.

The simulations are supposed to be interactive allowing stakeholders to participate.

SUD components. For each usage process, POSE allows to define the required resources and the set of software components maintaining these resources. The SUD is represented via the particular system of these components. For each version of the component, its representation is registered with POSE and made accessible to the stakeholders. It can be: (1) its requirements-based semantic model (QAPM representation of its functionality and initial qualities); (2) the set of its processes retrieved from PA; (3) its software prototype; (4) its final implementation.

POSE usage. One of the initial tasks the POSE users need to solve is the definition of the scope of the intended simulation, i.e., they need to define what counts as the organization under scrutiny, its structure, and the roles of its staff. After that, it is

necessary to specify its business processes and register SUD component models and prototypes. Then the users provide all data they need for conducting simulation experiments. This process is called *POSE parameterization*; it covers typical load data such as occurrence figures, availability of resources, probability of disasters etc.

After the processes are defined and the parameter data is specified, POSE runs the simulations of the usage processes. Stakeholders interact with them using the “business game” interface. When this interaction entails querying a SUD, POSE simulates the model of registered SUD component or tries out its prototype or final version. POSE users can make comments on the perceived system qualities or assess registered SUD components in formalized ways (e.g. via ranking a SUD version on some scale, or via pairwise comparisons of SUD versions). The places in the model where these assessments can be made are defined using QAPM crosscutting support. After that, requirements engineers analyze the requirements elicited out of these assessments. The semantics of these requirements is also represented using QAPM.

Usage example. Suppose we plan to use the proposed environment to elicit the response time requirement related to the *Account* actions in a banking system. First, we need to describe the business processes going on in a bank (in particular *Opening an account*) and define user roles in these processes (e.g. *Bank clerk*). These descriptions can be coded in any PML supported by POSE. Then, it is necessary to describe process models of the necessary SUD components (in particular, *Account management*) and register these models with the usage processes. In our case, for *Opening an account* process, the *Account management* component is registered to handle such actions as *Validate account information* etc. Then, the initial load figures (e.g. expected user load, hardware capacity etc.) are entered into the system (they will later affect the simulation). After that, the usage processes are interactively simulated and the stakeholders (e.g. real bank clerks) encounter a “business game” interface. In a process of interacting with POSE through this interface, stakeholders initiate requests handled by *Account management* component. If such request is issued, this component provides an answer with some simulated response time (which depends on load figures) and the stakeholder can assess it (e.g. using 1...10 scale). This assessment together with the simulated response time is then used to form an *elicited requirement*. Its QAPM representation is then made available to a requirement engineer.

5 Work to Do

Although the POSE approach is promising, some potential problems have to be solved, e.g. (1) the cost-benefit ratio of the approach might be prohibitive; (2) the POSE set up might be too difficult to do. Case studies are planned to find evidence in favor or against these claims.

References

1. Baniassad, E., Clarke, S.: Theme: An Approach for Aspect-Oriented Analysis and Design. In: Proc. ICSE 2004, pp. 158–167. IEEE CS Press, Los Alamitos (2004)
2. Barbacci, M., Ellison, R., Lattanze, A., Stafford, J., Weinstock, C.B., Wood, W.G.: Quality Attribute Workshops (QAWs), 3rd edn. Technical Report, CMU (2003)

3. Chitchyan, R., Sampaio, A., Rashid, A., Sawyer, P., Khan, S.: Initial Version of Aspect-Oriented Requirements Engineering Model. AOSD-Europe project report D36 (2006)
4. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Boston (2000)
5. Cleland-Huang, J., Settimi, R., Zou, X., Solc, P.: Automated Classification of Non-Functional Requirements. *Requirements Engineering* 12, 103–120 (2007)
6. Cysneiros, L.M., Yu, E.: Non-Functional Requirements Elicitation. In: *Perspectives on Software Requirements*, pp. 115–138. Kluwer Academic Publishers, Boston (2004)
7. Egyed, A.: Dynamic Deployment of Executing and Simulating Software Components. In: Emmerich, W., Wolf, A.L. (eds.) *CD 2004. LNCS*, vol. 3083, pp. 113–128. Springer, Heidelberg (2004)
8. Floyd, C.: A Systematic Look at Prototyping. In: Budde, R., Kuhlenkamp, K., Mathiassen, L. (eds.) *Approaches to Prototyping*, pp. 1–17. Springer, Berlin (1984)
9. Glinz, M.: On Non-Functional Requirements. In: *Proc. RE 2007*, pp. 21–26. IEEE CS Press, Los Alamitos (2007)
10. Harel, D., Pnueli, A., Lachover, H., Naamad, A., Politi, M., Sherman, R., Shtull-Trauring, A., Trakhtenbrot, M.: STATEMATE: A Working Environment for the Development of Complex Reactive Systems. *IEEE Trans. Soft. Eng.* 16, 403–414 (1990)
11. Heitmeyer, C.: Formal Methods for Specifying, Validating, and Verifying Requirements. *Journal of Universal Computer Science* 13, 607–618 (2007)
12. Hochmüller, E.: Towards the Proper Integration of Extra-Functional Requirements. *The Australian Journal of Information Systems* 7, 98–117 (1999)
13. iRise tool, <http://www.irise.com>
14. ISO 9000:2005, *Quality Management Systems - Fundamentals and Vocabulary* (2005)
15. ISO/IEC 9126-1, *Software Engineering – Product Quality – Part 1: Quality Model* (2001)
16. Kop, C., Mayr, H.C.: Conceptual Predesign – Bridging the Gap between Requirements and Conceptual Design. In: *Proc. ICRE 1998*, pp. 90–100. IEEE CS Press, Los Alamitos (1998)
17. Mayr, H.C., Bever, M., Lockemann, P.C.: Prototyping Interactive Application Systems. In: Budde, R., Kuhlenkamp, K., Mathiassen, L. (eds.) *Approaches to Prototyping*, pp. 105–121. Springer, Berlin (1984)
18. Serena Composer tool, <http://www.serena.com/products/composer/>
19. Seybold, C., Meier, S., Glinz, M.: Scenario-Driven Modeling and Validation of Requirements Models. In: *Proc. SCESM 2006*, pp. 83–89. ACM Press, New York (2006)
20. Shekhovtsov, V., Kaschek, R., Zlatkin, S.: Constructing POSE: a Tool for Eliciting Quality Requirements. In: *Proc. ISTA 2007*, GI, Bonn. LNI, vol. P-107, pp. 187–199 (2007)
21. Zlatkin, S., Kaschek, R.: Towards Amplifying Business Process Reuse. In: Akoka, J., Liddle, S.W., Song, I.-Y., Bertolotto, M., Comyn-Wattiau, I., van den Heuvel, W.-J., Kolp, M., Trujillo, J., Kop, C., Mayr, H.C. (eds.) *ER Workshops 2005. LNCS*, vol. 3770, pp. 364–374. Springer, Heidelberg (2005)

Classifying Assumptions Made during Requirements Verification of Embedded Systems

Jelena Marinčić*, Angelika Mader, and Roel Wieringa

Department of Computer Science, University of Twente, The Netherlands,
P.O.Box 217, 7500 AE Enschede, The Netherlands
`{j.marincic,mader,roelw}@ewi.utwente.nl`

Abstract. We are investigating ways to improve the process of modelling of embedded systems for formal verification. In the modelling process, we make a mathematical model of the system software and its environment (the plant), and we prove that the requirement holds for the model. But we also want to have an argument that increases our confidence that the model represents the system correctly (with respect to the requirement). Therefore, we document some of the modelling decisions in form of a list of the system assumptions made while modelling. Identifying the assumptions and deciding which ones are relevant is a difficult task and it cannot be formalized. To support this process, we give a classification of assumptions. We show our approach on an example.

1 Introduction

Models have increasing relevance in embedded system design. Our focus is on the construction of embedded systems verification models. Our goals are:

(1) We want to develop a modelling method. We share the observation of [1] that more research is spent on developing new languages and tools than on providing methods for using the existing ones. A major difficulty here is that modelling cannot be purely formal. We claim that the non-formal steps do not follow unpredictable irrationalism, but are part of educated creativity, following a systematic way of thinking.

(2) Having constructed a verification model we also want its justification - a correctness argument that makes us convinced that successful verification of the model reflects the desired behaviour of the embedded system. The correctness argument includes the assumptions and modelling decisions about the embedded system we have taken during modelling. Changing the assumptions can invalidate the model justification. Therefore, we propose to write down a list of the assumptions made while modelling.

(3) Identifying an assumption and deciding whether it is relevant are informal activities, difficult to capture by a formal approach. To help the modeller, we present a classification of assumptions. The classification presented in this paper

* Supported by the Netherlands Organisation for Scientific Research (NWO), project 600 065 120 241420, Modelling Control Aspects of Embedded Systems.

does not depend on a formal modelling or verification technique. The classifications of assumptions also help us understand the modelling activity itself. We believe that checking the assumptions we made against the classes we identified, gives more structure to the way of thinking and argumentation.

Terminology and Basic Concepts. An embedded system consists of a controller and a controlled, physical part. By **plant** we denote the controlled, physical part, and by **environment** everything outside the embedded system. The control software is abbreviated to **control**.

A **model** is a formal representation of the system, e.g., a diagram or a timed automaton. We model both the plant and the control and verify them against the required behaviour. The verification problem is to prove that a plant P and a control C together satisfy certain requirements R , denoted by $C \wedge P \models R$. This is analogous to [2, 3], but different from other approaches, where only the control software is modelled.

To conclude that the real system satisfies the required behavior, we need a **model justification** - an argument that the model and the formal requirement represent the system and the required behaviour. Such a justification can be given by reconstructing the modeling process into a rational process. In this paper we focus on the role of assumptions in rationalizing the modeling process.

In Sect.2 and Sect.3 we will present our classification of assumptions and will demonstrate it on an example. After briefly discussing related work in Sect.4 we will draw conclusions in Sect.5.

2 Classification of Assumptions

We define an assumption as a statement that refers to the plant and environment, and is taken for granted to be true for the purpose of the model justification. As control specifiers, we place constraints on the control behaviour, but we cannot place constraints on the plant; we can only make assumptions on its behaviour. Assumptions can be stated formally - then they are part of the formal proof, or non-formally - in that case they are part of the justification argument. The first two classes below answer the question *what* the assumptions are describing. The next two are focusing on the criteria of their changeability. The third group of the classifications focus on the relevance for the system users.

C1: Assumptions about system components. The requirement we want to verify determines where we draw the border between the system and its environment and what system aspects we will describe in the model. After that, we decompose the system, describe each component and, if necessary, decompose further. When decomposing the system, we simultaneously decompose the requirement, where each sub-requirement should be satisfied by a system component, and all sub-requirements together should imply the original requirement.

We can decompose the system in many different ways. We can make a process decomposition, a decomposition to the physical components, functional decomposition etc. The components can be described through assumption-requirement pairs in the form $assum(i) \implies req(i)$, where req is the subrequirement we

found while decomposing the system. For example: "If the wire is not longer than 12m (assumption), then the signal strength is sufficient for correct transmission (requirement)".

C2: Assumptions about system aspects. A system aspect is a group of system properties, usually related to one knowledge domain. An embedded system has electrical, mechanical aspect etc. When designing the control and verifying the system requirement, we might need assumptions coming from these different knowledge domains. If, e.g., we are designing shut-down system procedure for an embedded system, we want to know the electrical characteristics like capacity and resistance of the circuit that delays power off, to calculate the time the procedure has to save the data.

C3, C4: Necessary and Contingent Assumptions. Depending on the context in which we use the system, some of the assumptions we take as true and do not consider them as changeable.

Natural laws, like for example physics formulas, are considered to be true. If we have a system with a conveyor belt that transports bottles from the filling place, we will assume that its users will put the conveyor belt on a horizontal surface. Some of the plant components can be described with *engineering formulas* which we do not doubt. For example, the signal transmission through fiber optic cable is described with formulas that precisely calculate optical signal properties.

Contingent truths on the other hand may change. There are some facts about the system for which we are not sure whether they will change or not. In practice, it often happens that we have the plant and start designing the control software as if the plant is fixed, whereas in practice components are replaced. For example, if we have a conveyor belt that has to move faster, we can replace the existing motor with a more powerful one. (Then, we would have to change some parameters in control law implemented by control software.) Another example is that the plant is fixed, but our knowledge about it is changed. A domain expert can provide an improved formula describing the system behaviour.

C5: Constraints on the Plant and Embedded System Environment

Some of the assumptions we make pose constraints on the plant and users. We cannot be sure in advance that they will be fulfilled. The best we can do is to list them and deliver them together with the system. These assumptions are not part of the model - they can be seen as a label on the 'delivery box' of the system. For example: "If the weight in the cabin is larger than 20 and less than 150kg, the lift will go to the floor determined by the button pressed in the cabin."

3 Example - The Lego Sorter

The Lego sorter is a PLC (Programmable Logic Controller)-controlled plant made of Lego bricks, DC motors, angle sensors and a colour scanner [4]. Bricks of two colours are stored in a queue. They enter a belt one after another, and possibly more than one brick is on the belt. The belt is moved by a motor. Bricks are transported by the conveyor belt to the scanner and further on to the sorter.

The scanner can distinguish a yellow, blue or no brick in front of it. Putting a brick of another colour in front of it would cause the scanner to enter into an unknown state. The sorter consists of two fork-like arms. Each arm can rotate a brick to one of the sides of the plant. Each sorter arm is controlled by its own motor and has its own rotation sensor that senses the angle of the arm. The starting angle is 0, and as the arm rotates it changes to 360 degrees.

Classification C1: Assumptions about components		Classification C2: Assumptions about aspects		Classification C3, C4: Necessary and contingent assumptions	
Components	Assumptions	Aspects	Assumptions		
Queue	A6: There are only blue and yellow bricks in the Queue. A7: The motor moving the Belt is working properly.	Control eng.	A3: The sampling period is such that control can observe rotation angle with sufficient granularity. A4.1: The Scanner transmits the signal with the delay D1 A9: The rotation sensors ...transmit signals with no delay. A9.1: The rotation sensors show only relative arm position.	Natural laws	If on the Belt for T1 seconds, and the Belt is moving, the brick changes its position. If the Belt is stopped, and the brick is on it, it won't change its position.
Belt	A12: There is a minimal distance between bricks so that A12.1: There is always "nothing_at_scanner" observed by Scanner before the new brick is observed and A12.2: There will be no new brick in front of the Scanner before a previous brick moves to the Sorter A13: The order of a brick observed at Scanner and a new brick entirely on the Belt is not deterministic. A15: The order of a brick arrival to the Scanner and previous brick sorted is not deterministic.			Engineering formulas	The Belt speed v is a function of the value m put on the Belt motor: $v = k \times m$
Scanner	A8: The Scanner observes the color all the time (not interrupt-driven). A8.1: The Scanner transmits the signal with the delay D1 A12, A13, A15	Mechanical	A1: Computer hardware works properly. A12: There is a minimal distance between bricks so that A12.1: There is always "nothing_at_scanner" observed by Scanner before the new brick is observed and A12.2: There will be no new brick in front of the Scanner before a previous brick moves to the Sorter. A13: The order of a brick observed at the Scanner and a new brick lying entirely on the Belt is not deterministic. A15: The order of a brick arrival to the Scanner and previous brick sorted is not deterministic. A16: Bricks are standard Lego bricks (50mm x 15mm x 7mm) that fit in the Queue. A17: The motor moving the Belt work properly.	Contingent truths	Here, all the descriptions about the Scanner are contingent. We might replace the Scanner that for example distinguishes more colours or that has a zero delay The same holds for other plant assumptions; we might, for example, reconstruct the Plant in such a way that the Sorter arms have to move in different direction when sorting
Sorter	A9: The rotation sensors and motors work properly and transmit signals with no delay. A9.1: The rotation sensors show only relative arm position. A11: The sorter starts with proper initial position. A12, A15 A14: Upon start, Sorter and Belt are empty.			Classification C5: Plant and environment constraints	
Bricks	A16: Bricks are standard Lego bricks (50mm x 15mm x 7mm) that fit in the Queue. A12, A13, A15	Electrical	A9: The rotation sensors and motors work properly A17: The sorter should start with proper initial position.	A5: An operator will put the bricks in the Queue. A6: There are only blue and yellow bricks in the Queue. (If a brick of any other colour is there, the Scanner will recognize it as Yellow or blue and will sort it on one of the sides.) A11: The sorter should start with proper initial position.	
Operator	A5: An operator will put the bricks in the Queue. A11: The sorter starts with proper initial position. A14: Upon start, Sorter and Belt are empty.	Implementation	A2: The operating system supports the control that we design. A4: The PLC controller supports the desired sampling frequency. A8: The Scanner observes the color all the time (not interrupt-driven).		
Controller hardware and OS	A1: Computer hardware works properly. A2: The OS supports the control we design. A3: The sampling period is such that control can observe rotation angle with sufficient granularity. A4: The PLC controller supports the desired sampling frequency.				

Fig. 1. List of the assumptions shown according to the classification criteria we found

The requirement is: “*Eventually all the bricks from the queue will be moved by the sorter to the side corresponding to their colour*”. We designed the control, and modelled and verified the system (see [5] and [6]). The assumptions that we identified are presented in table in Fig.1. Some of the assumptions are part of the model, but are also listed in the table.

4 Related Work

From the work following the approach of [2] we mention only the most similar to ours. The problem frames technique [3] defines *frame concerns* through examples of issues that have to be addressed and that are not described in the problem diagrams, e.g., initialization of the software and hardware.

In [7] a technique for software specification is described, starting from the requirement for the plant. Assumptions (‘breadcrumbs’) on the plant are collected, and an argument for each modelling step. No guidelines for finding assumptions are given.

In [8] a formal conceptual network based on problem-oriented perspective is developed, where modelling steps are formally described. We, on the other hand, are looking for ways to systematically perform these steps.

In the area of requirements engineering, the goal-oriented methods have a significant place. Our classification of assumption could be useful in the phases of requirements analysis of the KAOS method [9]. In the Tropos methodology [10], when defining the circumstances under which a given dependency among two actors arises, a modeller has to learn about the system, so our assumption classification might be useful there, too.

The problem of modelling method is addressed in [1] by *agendas*, a list of modelling steps. The transition from informal to formal is performed in one of the first steps of the requirements elicitation, while we formalize only the last steps when the complete knowledge about the system is available.

In [11] a General Property-oriented Specification Method is introduced, where assumptions are collected in the cells of a table made while decomposing the system. This framework is restricted to the use of labelled transition systems.

5 Discussion and Conclusion

Formal methods are applied in a non-formal world and we cannot give an algorithm how to collect the assumptions. Instead, we found different classes of assumptions that are made in the modelling process and different ways of identifying the assumptions.

Making assumptions explicit is not so much a matter of using the appropriate languages or tools. In the first place it requires a discipline of thought, and being aware what we do during modelling activity can help here by saying at which point of the modelling process we have to look for assumptions, and which form these can have. Different categories of assumptions mean that we have different views to the system, even if we chose one decomposition. If we restrict ourselves

into one single view or decomposition, we might omit an important assumption. Therefore, classification of assumptions is useful as a checklist to go through when describing the system; this is a hypothesis that needs further proving. An experiment in which a group of modellers will be presented with assumptions classification and one not, is needed to make this statement an empirical claim. This is the part of our further work.

We plan to look closer into subclasses of embedded control systems for which we can make specialized, more concrete modelling guidelines. We will focus on the communication of control engineers and verification experts while doing formal verification, to identify the boundaries of these two knowledge domains, and to make more clear what one expert has to know about other expert's area.

References

- [1] Heisel, M., Souquières, J.: A method for requirements elicitation and formal specification. In: Akoka, J., Bouzeghoub, M., Comyn-Wattiau, I., Métais, E. (eds.) ER 1999. LNCS, vol. 1728, pp. 309–325. Springer, Heidelberg (1999)
- [2] Zave, P., Jackson, M.: Four dark corners of requirements engineering. *ACM Trans. Softw. Eng. Methodol.* 6(1), 1–30 (1997)
- [3] Jackson, M.: *Problem Frames: Analysing and Structuring Software Development Problems*. Addison-Wesley, Reading (2000)
- [4] MOCA project - ongoing work, <http://moca.ewi.utwente.nl/WORK.html/>
- [5] Marinčić, J., Wupper, H., Mader, A., Wieringa, R.: Obtaining formal models through non-monotonic refinement. Technical report TR-CTIT-07-33, CTIT, Univ. of Twente, The Netherlands (2007)
- [6] Marinčić, J., Mader, A., Wieringa, R.: Capturing assumptions while designing a verification model for embedded systems. Technical report TR-CTIT-07-03, CTIT, Univ. of Twente, The Netherlands (2007)
- [7] Seater, R., Jackson, D., Gheyi, R.: Requirement progression in problem frames: deriving specifications from requirements. *Requir. Eng.* 12(2), 77–102 (2007)
- [8] Hall, J.G., Rapanotti, L., Jackson, M.: Problem oriented software engineering: A design-theoretic framework for software engineering. *sefm* 0, 15–24 (2007)
- [9] Dardenne, A., Fickas, S., van Lamsweerde, A.: Goal-directed concept acquisition in requirements elicitation. In: *Procs of IWSSD 1991*, pp. 14–21. IEEE Computer Society Press, Los Alamitos (1991)
- [10] Giorgini, P., Mylopoulos, J., Sebastiani, R.: Goal-oriented requirements analysis and reasoning in the tropos methodology. *Engineering Applications of Artificial Intelligence* 18/2 (2005)
- [11] Choppy, C., Reggio, G.: Towards a formally grounded software development method. Technical Report DISI-TR-03-35, Università di Genova, Italy (2003)

Integrating Portfolio Management and Simulation Concepts in the ERP Project Estimation Practice

Maya Daneva

University of Twente
m.daneva@utwente.nl

Abstract. This paper presents a two-site case study on requirements-based effort estimation practices in enterprise resource planning projects. Specifically, the case study investigated the question of how to handle qualitative data and highly volatile values of project context characteristics. We counterpart this challenge and expound upon the integration of portfolio management concepts and simulation concepts into a classic effort estimation model (COCOMO II).

1 Introduction

Business-requirements-based effort estimation is a practical part of the early stage of any Enterprise Resource Planning (ERP) project. Though, how to construct realistic schedules and budgets so that an ERP-adopting organization can achieve cost-effective and timely project delivery is, by and large, unknown. Researchers [3,10] indicate that existing project estimation practices (e.g. [1]) are limited by their inability to counterpart the challenges which the ERP project context poses to estimation analysts, for example, how to account for the uncertainties in cost drivers unique to the diverse configurations, system instances and versions [3] included in the solution. Here we explore one possible approach as a remedy to this situation. In case study settings, we complementarily deployed the COCOMO II effort estimation model [1] at the requirements stage, and portfolio management (PM) and Monte Carlo (MC) simulation concepts. In what follows, we provide a background on the approach and, then, we report on our case study plan, its execution, and our early conclusions.

2 Background

Our approach to uncertainties in ERP effort estimation rests on four types of sources: (i) the COCOMO II model [1] that lets us account for ERP adopter's specific cost drivers, (ii) the MC simulation [8] which lets us approach the cost drivers' degrees of uncertainty, (iii) the effort-and-deadline-probability-based PM concept [9] which lets us quantify the chance for success with proposed interdependent deadlines for a set of related ERP projects, and (iv) our own experience in ERP RE [4], which was used to incorporate the effort estimation process into the larger process of early RE (that is, at time of bidding). We chose the combination of (i), (ii) and (iii), because other researchers already used it [7] and found it encouraging. In marked contrast with these authors [7] who blended these techniques for the purpose of custom software

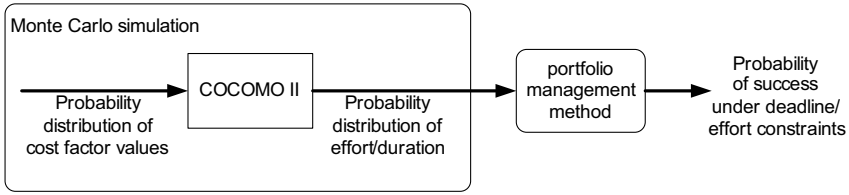


Fig. 1. A conceptual model of the solution approach

contract bidding, we adapt the techniques to the ERP project context and we use them jointly therein.

Fig. 1 presents how the three techniques fit. Because we want to incorporate ERP project context uncertainties into the project estimates, we suggest COCOMO II take as inputs the probability distributions of the COCOMO scale factors and cost drivers (instead of taking as inputs single values as in [1]). We use the MC simulation to get randomly-selected values into COCOMO II and, then, see how likely each resulting outcome is. Our approach yields as a result the possible effort and duration estimation values for each uncertain factor. Unlike COCOMO II, our output is the probability distributions of effort and duration and not the most likely effort and duration which COCOMO II creates. The probability distributions are fed into the PM method [9]. To run it, we first bunch projects into portfolios and, then, obtain the probability of successfully delivering the projects under both time and effort constraints. The application of this solution in context is described below.

3 The Case Study Plan

Our case study was planned as per the guidelines in [14]. Our overall goal was to determine whether the use of PM increases the chance of success and, if so, to what extent. Our expectation was that the ERP projects with high uncertainty ratings of the COCOMO II scale factors and cost drivers would benefit more from PM, than the projects with low uncertainty ratings would. The scope of the case study covers two sites in a large North-American company. Each site represents an independent business unit running their own ERP projects based on a specific package. Prior to the case study, the units were independent firms which merged. While the first site implemented three modules of the PeopleSoft ERP package, the second site [5] rolled out a large organization-wide ERP solution that included eight functional modules of the SAP package. A condensed summary of the case study setup pertaining to the SAP site has been presented in a ESEM'07 poster [5].

The three techniques from Fig. 1 are summarized as follows:

COCOMO II: We used it because (i) it's a popular and comprehensive empirical parametric model [1] and (ii) both our sites had data allowing its use. COCOMO II produces estimates of effort and duration by using two equations as follows:

$$\text{Effort} = A \times (\text{Size})^E \times \prod_{i=1}^{17} EM_i \quad \text{and} \quad \text{Duration} = C \times (\text{Effort})^F \quad (1)$$

Therein, E and F are calculated via the following two expressions, respectively:

$$E = B + 0.01 \times \sum_{j=1}^5 SF_j \quad \text{and} \quad F = D + 0.2 \times (E - B) \quad (2)$$

In (1), 17 cost drivers (EM) serve to adjust initial effort estimations. In (2), five scale factors (SF) reflect economies and diseconomies of scale observable in projects of various sizes. The degrees of these 22 context characteristics are rated on a seven-point scale, ranging from ‘extra low’ to ‘extra high’.

Monte Carlo simulation: This is a problem-solving technique to approximate the probability of certain outcomes by running multiple trial runs, called simulations, using random variables. Here, we use it to obtain a range of possible values for our estimates, while taking the COCOMO II cost drivers and their degrees of uncertainty as inputs. We borrowed this idea from the THAAD Project Office [8] and the JLP NASA [6]. We run it according to these steps: (1) ascribe a particular distribution type to an input variable in COCOMO II; (2) repeatedly run the model 10000 times and collect samples of the output variables for each run so that we produce an overall picture of the combined effect of different input variables distribution on the output of the model; (3) plot a histogram showing the likelihood of obtaining certain output values for the set of input variables and attached distribution definitions.

Portfolio management: The PM method in [9] quantifies the uncertainty associated with a project estimate for a set of projects managed as a portfolio. It gives the opportunity to obtain a probability of a portfolio’s success under effort and schedule constraints. We chose it because: (i) it is applicable at the stage of requirements [9], (ii) its only input requirement is a record of previous projects; and (iii) it fits with the ERP adopters’ project realities suggesting that an ERP project is implemented as a portfolio of interdependent subprojects. Each subproject is a piece of functionality (or an ERP module) linked to other pieces. For example, the Asset Management functionality in a package is tightly linked with the Financial Accounting module and the Controlling module. Given a set of interdependent subprojects, the effort estimation model yields (i) the probability of portfolio’s success with the proposed deadlines for each subproject in this portfolio, and (ii) a set of new deadlines which will result in a required probability of success. The portfolio success is judged by two conditions applied to any two subprojects a and b for which deadline_a is earlier than deadline_b . The conditions are that: (i) subproject a is to be over by deadline_a and (ii) subproject a and subproject b are to be over by deadline_b . In other words, the conditions require all subprojects planned with a deadline before deadline_b to be completed by deadline_b , rather than just project b . This is the key to the portfolio approach, because uncertainty about completion of project b incorporated uncertainty from all previous projects. Suppose in total E people are on the project and let d be the number of work days it takes from start date to deadline, then the total available resources is Exd . So, suppose an ERP portfolio Y is made up by n subprojects, the success conditions are represented as follows:

$$\begin{pmatrix} Y_1 \\ Y_1 + Y_2 \\ \dots \\ Y_1 + Y_2 + \dots + Y_n \end{pmatrix} \leq E \begin{pmatrix} d_1 \\ d_2 \\ \dots \\ d_n \end{pmatrix} \quad (3)$$

where Y_i is the estimated effort for subproject i to succeed. We check if, for any j , ($j=1..n$), the sum of Y_1, \dots, Y_j is greater of Exd_j . If this is true, then deadline d_j has failed. Success probabilities result from simulations in which Y_1, \dots, Y_n are generated from a predetermined probability distribution. If we deem Y_1, \dots, Y_n is satisfying all conditions, then we say that the portfolio Y succeeds. The portfolio's probability of success is equal to the ratio of the number of successes in the set Y to the number of trials in the simulation.

4 Case Study Execution

We modeled the uncertainty of the 22 context factors by means of a probability distribution, which means identifying for each factor (i) its distribution type and (ii) its parameters. We did this based on proposed default choices by other authors [6,7,8], e.g. McDonald's [8] default 'high' levels of uncertainty associated to the ratings of the RESL, DATA, ACAP and PCAP cost drivers [1]. The level of uncertainty determines, in turn, the distribution type to be assigned to each cost driver: normal, triangular, and uniform for low, medium and high uncertainty, respectively.

Next, the matter that COCOMO II provides duration estimation (2), encouraged us to formulate the following condition for PM in terms of time constraints:

$$\begin{pmatrix} T_1 \\ T_1 + T_2 \\ \dots \\ T_1 + T_2 + \dots + T_n \end{pmatrix} \leq \begin{pmatrix} m_1 \\ m_2 \\ \dots \\ m_n \end{pmatrix} \quad (4)$$

where T_i is the ERP implementation time in months for subproject i . We note that this condition does not include the number of people E , because COCOMO II assumed an average number of project staff [1] which was accounted in (2). Furthermore, as recommended in [7], we attempted to improve the chances for portfolio success by adjusting the cost drivers and scale factors. Hence, we adopted the assumption that for projects with two different ratings for the same factor, the probability of success for each project will be different too.

Project data: The data we used in the first site were collected from six PeopleSoft projects completed between May'98 and June'00 and the data in the second site - from 13 SAP projects carried out between Nov'97 and Oct'03. In this period, the author was employed by the case company as a SAP process analyst and was actively involved in the projects. In both sites, for each project, we got (i) project size data, (ii)

reuse levels, (iii) start and end dates, and (iv) scale factor and cost driver ratings. Size (see equation (1)), was measured in terms of unadjusted Function Points (FP) [4]. We counted it by using the standard rules of the International Function Point User Group (IFPUG, www.ifpug.org). The first site employed a IFPUG-certified FP specialist who counted FP from requirements and architecture design documents delivered by the PeopleSoft consultants on board. The second site also followed the IFPUG standard, but used the counting rules specifically refined to the observable elements of functionality in the SAP business requirement documents [4]. The effort multipliers A , B , and EM , and the scale factors SF were calibrated for each site by using ERP effort data collected between 1997 and 2004 in the two business units. We note that in both sites, we did not have any knowledge about the uncertainty of the scale factors and cost drivers ratings and therefore, we used default levels proposed by other authors [6,7,8]. We opted to use a lognormal distribution for functional size, as this was motivated by Chulani's observations [2] that (i) the skew of the size distribution is positive and that (ii) $\log(\text{size})$ is likely to be a normal distribution. With this input data, we run MC simulations (a total of 10000 trials) which gave us samples of (i) effort, expressed in person-months, and (ii) duration, expressed in months.

Results: To see how the change of uncertainty levels of a cost driver rating impacts the project success under effort and schedule constraints, we constructed two portfolios: the first one had this driver rated as 'very high' for all projects and the other portfolio had it rated as 'very low' for all projects. For each portfolio, we calculated the probability of success under time constraints and under effort constraints. For example, Table 1 indicates that – at both sites, when the ERP-specific tools (TOOL [1]) were used in the project, the probability of success was higher under both time and effort constraints.

Table 1. Analysis of the probability of success for the factor TOOL under effort constraints

TOOL rating	Site	Probability of success	
		Under effort constraints	Under time constraints
Very low	PeopleSoft	46.33%	51.54%
Very high	PeopleSoft	97.99%	96.88%
Very low	SAP	49.27%	51.88%
Very high	SAP	98.01%	95.72%

Table 2. Probability of success for low/high uncertain projects under time constraints

Uncertainty level	Site	Probability of success		Ratio of increase (b)/(a)
		Individual projects (a)	Portfolio (b)	
Low uncertainty	PeopleSoft	21.45%	90.93%	4.23
High uncertainty	PeopleSoft	14.31%	86.77%	6.06
Low uncertainty	SAP	15.76%	87.52%	5.55
High uncertainty	SAP	8.31%	75.91%	9.13

In both sites, we observed that 13 of the 17 COCOMO II drivers can be adjusted in a way that maximizes the chance of success. Furthermore, we used ‘the ratio of increase’ [5,7] (i.e. the utmost right column in Table 2) to see whether the probability of success increases (and if so by how much) when projects are managed as a portfolio. Table 2 suggests that bundling ERP projects as a portfolio had the advantage over managing projects separately under time constraint.

5 Conclusions

Many issues arise when estimating ERP project costs from early requirements. This two-site case study applied an approach targeted to resolve the issue of volatile values of context factors which impact project outcomes. We learnt that: (1) to get a better estimate, we must be flexible and open enough to exploit the power of synergies among the three techniques and to learn from qualitative details of context; (2) examining the uncertainties in the context of each ERP portfolio clearly and from diverse sides helps us learn more about the effort estimation problem we face; (3) to ERP-adapters, this approach might be one good alternative over vendor-provided project estimates. However, our results are preliminary only. We are aware of related validity concerns [11] and plan a series of case studies to test our approach, to properly evaluate its validity and to come up with an improved version of it.

References

- [1] Boehm, B.: Software Cost Estimation with COCOMO II. Prentice Hall, Upper Saddle River (2000)
- [2] Chulani, S., Boehm, B.W., Steece, B.: Bayesian Analysis of Empirical Software Engineering Cost Models. *IEEE Trans. on Software Eng.* 25, 573–583 (1999)
- [3] Daneva, M., Wieringa, R.: Cost Estimation for Cross-organizational ERP Projects: Research Perspectives. *Soft Quality J.* 16 (2008)
- [4] Daneva, M.: Measuring Reuse of SAP Requirements: a Model-based Approach. In: 5th International Symposium on Software Reusability, pp. 141–150. ACM Press, LA (1999)
- [5] Daneva, M.: Approaching the ERP Project Cost Estimation Problem: an Experiment. In: 1st International Symposium on Empirical Software Engineering and Measurement, p. 500. IEEE Press, New York (2007)
- [6] Hihn, J.: Model-based Estimate. Technical report, JLP NASA (2004)
- [7] Jiamthubthugsin, W., Sutivong, D.: Portfolio Management of Software Development Projects Using COCOMO II. In: 34th IEEE International Conference on Software Engineering, pp. 889–892. IEEE Press, New York (2006)
- [8] McDonald, P., Giles, S., Strickland, D.: Extensions of Auto-Generated Code and NOSTROMO Methodologies. In: 19th International Forum on COCOMO, LA (2004)
- [9] Fewster, R.M., Mendes, E.: Portfolio Management Method for Deadline Planning. In: 9th International Software Metrics Symposium, pp. 325–336. IEEE Press, Los Alamitos (2003)
- [10] Stensrud, E.: Alternative Approaches to Effort Prediction of ERP Projects. *J. Inf. Soft. Techn.* 43, 413–423 (2001)
- [11] Yin, R.: Case Study Research, Design & Methods. 3rd edn. Sage, Newbury Park (2002)

Can Patterns Improve i* Modeling? Two Exploratory Studies

Markus Strohmaier¹, Jennifer Horkoff², Eric Yu³, Jorge Aranda²,
and Steve Easterbrook²

¹ Knowledge Management Institute, Graz University of Technology and Know-Center,
Inffeldgasse 21a, Graz, Austria

² Department of Computer Science, University of Toronto, 10 King's College Road,
Toronto, Ontario, Canada

³ Faculty of Information Studies, University of Toronto, 140 St. George St., Toronto,
Ontario, Canada

markus.strohmaier@tugraz.at, yu@fis.utoronto.ca,
{jenhork, jaranda, sme}@cs.utoronto.ca

Abstract. A considerable amount of effort has been placed into the investigation of i* modeling as a tool for early stage requirements engineering. However, widespread adoption of i* models in the requirements process has been hindered by issues such as the effort required to create the models, coverage of the problem context, and model complexity. In this work, we explore the feasibility of pattern application to address these issues. To this end, we perform both an exploratory case study and initial experiment to investigate whether the application of patterns improves aspects of i* modeling. Furthermore, we develop a methodology which guides the adoption of patterns for i* modeling. Our findings suggest that applying model patterns can increase model coverage, but increases complexity, and may increase modeling effort depending on the experience of the modeler. Our conclusions indicate situations where pattern application to i* models may be beneficial.

Keywords: The i* Framework, Model Patterns, Modeling Effort, Model Coverage, Model Complexity.

1 Introduction

In the field of requirements engineering, much work has been dedicated to modeling in the early stages of the requirements engineering process. Models created in the i* Framework capture the goals of stakeholders and help requirements engineers to understand the strategic interactions and dependencies among agents [20]. These models are assumed to, for example, facilitate analysis and discover new knowledge about the domain. However, widespread adoption of such models in the requirements engineering process has been hindered by a series of issues [7], including:

Costs of modeling: The effort necessary to create, maintain, understand, and analyze i* models is high.

Model coverage: Due to the high complexity of social relations, i* models may fail to cover all relevant issues.

Complexity of models: At the same time, the models that result from modeling with the i* Framework can be complex and difficult to scale.

Improving some of these aspects would represent an improvement to i* modeling practices. Usage of patterns in previous work suggests that patterns in general can provide, among others, the following benefits ([1],[3],[4]):

Reuse: By abstracting and packaging domain knowledge in a structured way, patterns enable the reuse of knowledge.

Modularization: Because patterns have a clearly defined focus and well defined areas of application, they contribute to modularizing the domain.

Communication: By providing an agreed upon vocabulary of domain knowledge, patterns facilitate communication among stakeholders.

Although there have been some initial efforts in using patterns for agent-oriented, social focused modeling ([15], [17]), patterns have not yet been applied extensively in this area. This might be because the use of patterns in this area brings new challenges related to pattern construction, selection, adaptation and evaluation, whose effects might cancel out the benefits of pattern application. Our research sets out to explore challenges related to patterns in an early requirements context.

Of the many different types of patterns which can be constructed, we are especially interested in the utility of model patterns, that is, patterns that capture knowledge for reuse in the form of conceptual models rather than textual descriptions. Specifically, we define i* model patterns as i* models which are generalizations of a particular domain or situation of interest, which can then be contextualized when applied to a more specific situation. In this work we focus on those patterns which describe the roles and intentions involved in the use of specific software or technologies in an abstract and reusable way. We focus on these types of patterns because they relate to the challenges and typical use of i*, specifically, enabling the evaluation of a particular technological solution in a specific context. To acquire a deeper understanding about the effects of pattern use in this context we have raised a set of research questions and conducted both an exploratory case study and an initial experiment involving the construction, application and evaluation of i* model patterns. A methodology for i* pattern application is introduced as part of an exploratory case study in Section 5.

2 Patterns in the i* Framework

Although the application of patterns shows promise in addressing several of the issues associated with i* modeling, i* model patterns as we have defined them differ from patterns typically seen in later stage requirements and software engineering. These differences often involve form (textual versus graphical representation) and focus, with i* models focusing on high-level solutions in the early stages of requirements analysis. In the area of requirements engineering, patterns have been used to capture

and organize knowledge about requirements and requirements engineering techniques. In software engineering, pattern theory defines a pattern as a construct that captures some proven knowledge of a domain via problem/context/solution triples which are created for further reuse [1]. In contrast, model patterns, as we define them, focus on the social context and interactions of the pattern subject matter. The patterns used in this work capture general requirements for the technology in the form of i* elements as well as general goals of the roles it interacts with, including the dependencies between them.

We assume familiarity with the i* Framework [21]. As an example of an i* model pattern, consider a pattern describing the social relationships surrounding the usage of a wiki, as shown in Fig. 1. We expect that particular instances of wikis (expressed as *contextual models*) would have many of the features depicted in this model pattern, but possibly also deviations from it.

The wiki, as a technology system, is modeled as an agent. Its main task is to Provide for Mass Collaborative Authoring. It exists in the context of a number of roles – visitors, editors, reviewers, as well as a, Technology “Champion”, who wants to promote the benefits of the wiki. The champion depends on the wiki to achieve the goal Content be Correct/Useful as part of facilitating Collaborative Authoring. Each actor (agent or role) has its own goals and tasks and softgoals (success criteria), but ultimately depend on each other to form a socio-technological system.

In this example, and in our pattern application methodology, the check marks in the model are used to indicate the extent to which the actor’s goals are achieved,

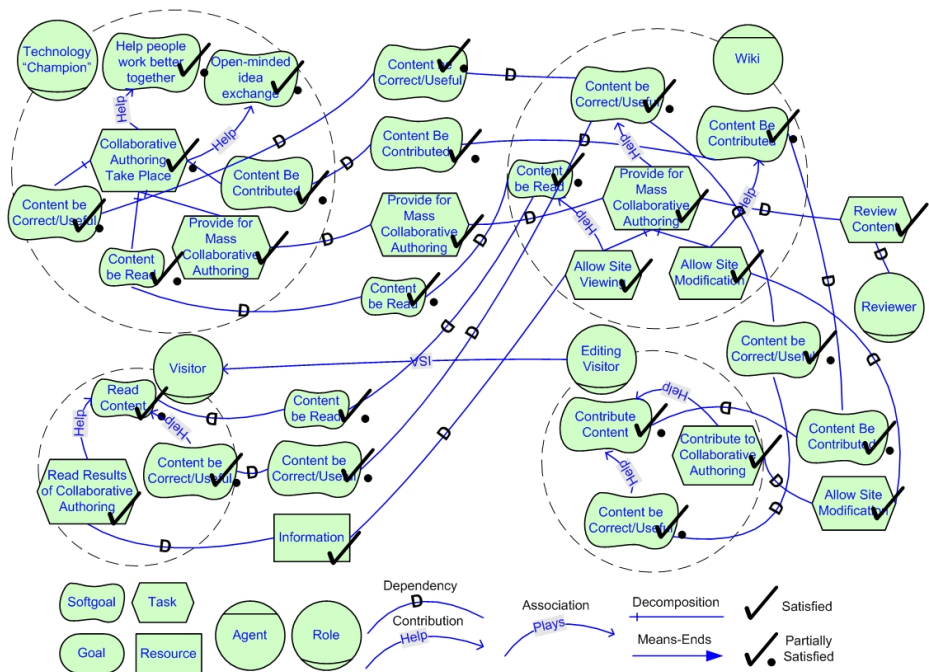


Fig. 1. Simplified Version of the Wiki Pattern

using a procedure described in [13]. Generally, the leaf elements or operationalizations in the model are marked as satisfied, assuming their implementation. These values are propagated throughout the model using a combination of automatic rules, based on the semantics of the links, and human intervention, to resolve conflicting or partial evidence. The results of the propagation are analyzed to determine if the needs of all actors are sufficiently met.

3 Research Questions and Research Design

Claims that the adoption of patterns improves requirements engineering efforts are abundant in literature, but the actual - positive or negative - effects that patterns have on requirements engineering in general and *i** modeling in particular have not been studied in detail. Therefore, the overarching question of this paper is "*Can patterns improve modeling with the i* Framework?*" In order to clarify what we mean by "improve" and to make this question amenable to scientific investigations, we formulate a set of more specific questions. This work does not aim to find definite answer to these questions, but instead aims to find evidence which begins to support or deny our preliminary claims.

Q1: Do model patterns help reduce modeling effort? Because *i** model patterns are designed with reuse in mind, model patterns should contribute to decreasing the effort involved in *i** modeling.

Q2: Do model patterns help increase model coverage? By capturing and documenting deep domain knowledge, the utilization and combination of *i** model patterns should increase the degree to which *i** models cover relevant aspects of the world.

Q3: Do model patterns help decrease complexity? Because *i** model patterns have a scope and clearly defined borders, they should help to make the high complexity of *i** models more manageable through modularization.

In order to investigate the positive and/or negative effects of patterns on *i** modeling we need to observe instantiations of the modeling process. For this reason, we employ a research approach which uses a case study as well as an exploratory experiment to study the introduced research questions. The case study involves an ongoing requirements analysis project with an external organization, while the follow-up experiment, designed to address some of the limitations of the initial case study, uses student participants in a classroom setting.

4 Case Study: Kids Help Phone

In order to investigate our research questions concerning the use of patterns, we developed a methodology, or series of concrete steps, that guides and constrains the application of model patterns to *i** models. In this section we outline the general steps of our proposed methodology, provide a description of the execution of these steps in the Kids Help Phone (KHP) case study, and present selected results.

4.1 Case Study Context and Preparation

This study uses data from an ongoing requirements analysis project with a not-for-profit youth counseling organization. KHP is a charitable, non-governmental organization that provides 24/7 counseling to kids across Canada via phone and web. The project was aimed to explore the situational "effectiveness" of a range of social technologies, such as discussion forums and wikis in their operations. To create an empirical baseline for our investigations, we interviewed stakeholders at KHP and constructed i* models of the domain without focusing on patterns. Specifically, we interviewed a total of ten stakeholders on their issues with knowledge transfer, in interview sessions lasting approximately one hour. The interviews acted as a basis for creating models that focused on the current usage of different technologies, such as a discussion forum. Finally, we assessed the current situation of KHP by evaluating the created models as a baseline for analyzing alternative solutions. In the case study, we chose to focus on a model representing the usage of a discussion forum.

4.2 Methodology and Case Study Execution

The left side of Fig. 2 provides a high-level overview of the steps involved in our methodology: from pattern creation, insertion and integration to the final evaluation of the resulting model. The right side of this figure contains some corresponding quantitative results of the study, explained in future sections.

4.2.1 Pattern Creation

This step involves the creation of i* model patterns. This will not be necessary once a patterns catalog becomes available.

Create Patterns. Create a set of patterns by consulting relevant literature. Model the roles, goals, tasks, resources, dependencies and contribution links related to a specific technology.

Case Study Application. In order to be able to evaluate a pattern approach in our case study, we created two model patterns – one pattern containing the use of wikis and one containing a discussion forum (Disc. F.). We applied the first pattern in a case where the original technology in the domain (a discussion forum) is replaced by a new technology (a wiki), and applied the second pattern in a case where a model of an existing technology is replaced by a more detailed, generalized model of this technology.

Evaluate Patterns. Evaluate the model patterns, (using the qualitative procedure described in [13]), in order to ensure that the goals of the pattern are, in principle, achievable in certain scenarios.

Case Study Application. Both the wiki and discussion forum patterns were evaluated in light of various common implementation scenarios. See Fig. 1 for a simplified version of the wiki pattern containing an evaluation of stakeholder goals.

4.2.2 Pattern Application

1. Select Patterns. Select patterns which are believed to be applicable and beneficial in the contextual model. Compare the contributions of goals in the pattern to the goals expressed in the contextual model(s) for an indication of pattern applicability.

Case Study Application. In the case of KHP, we chose the two patterns we had previously created.

2. Contextualize the Pattern. View the selected pattern in light of the contextual model domain, adding and removing relevant and irrelevant links and elements.

Case Study Application. We contextualized the wiki model pattern, removing 7 of the 117 elements and 16 of the 169 links. In the case of the discussion forum model pattern, all elements and design options were considered relevant and no changes were made.

3. Insert the Pattern. Insert the pattern into the contextual model view.

Case Study Application. In each case, the model pattern was pasted into the contextual model file containing the discussion forum.

4. Linking Actors. Link the actors defined in the pattern to the actors in the contextual model.

Case Study Application. In the case of the wiki model pattern, we replaced the discussion forum of the contextual model with the wiki model pattern. The pattern contained roles such as Visitor, Editing Visitor or Technology “Champion”, which were linked to the existing roles in the contextual model via the i* Framework’s actor association links (such as “PLAYS” and “IS-A”).

5. Pattern Integration. Integrate the pattern into the contextual model.

Case Study Application. The interactions between the pattern and the contextual actors were considered by adding or changing existing dependency links. The domain actors depended on their new roles in order to satisfy their goals, and, conversely, the technology agent depends on these actors, possibly indirectly, to be successful. In addition, we changed the existing elements and links in order to connect the new technology to the goals of existing actors.

We use measurements of model size and model changes as a way of quantifying our observations in relation to our research questions, see “Threats to Construct Validity” in Section 8 for a discussion of these measurements. As the first two points of measurement (“CM“, “MP”), we considered the size of the contextual model and model patterns before pattern insertion and integration. As the third point of measurement (“CIP”) we considered the size of the model after all pattern integration changes were made. During the integration process, the number of i* constructs added, deleted or changed in some way was recorded. A summary of the measurements appears in the table on the right side of Fig. 2. Note that the differences between the top and bottom size counts in the table do not balance with the changes reported in the middle, as the measurements for steps 2 and 3 are not reported.

6. Evaluate Model. Evaluate and analyze the resulting model to determine whether or not the technology represented by the model pattern is successful, both in terms of its own goals and the goals of the contextual actors. Compare the results with the evaluation of the existing technology in the contextual model.

Case Study Application. In the case of the wiki pattern, two possible wiki configurations were evaluated, one with periodic reviewing of content and one where content must be reviewed before being posted to the wiki. Although the second

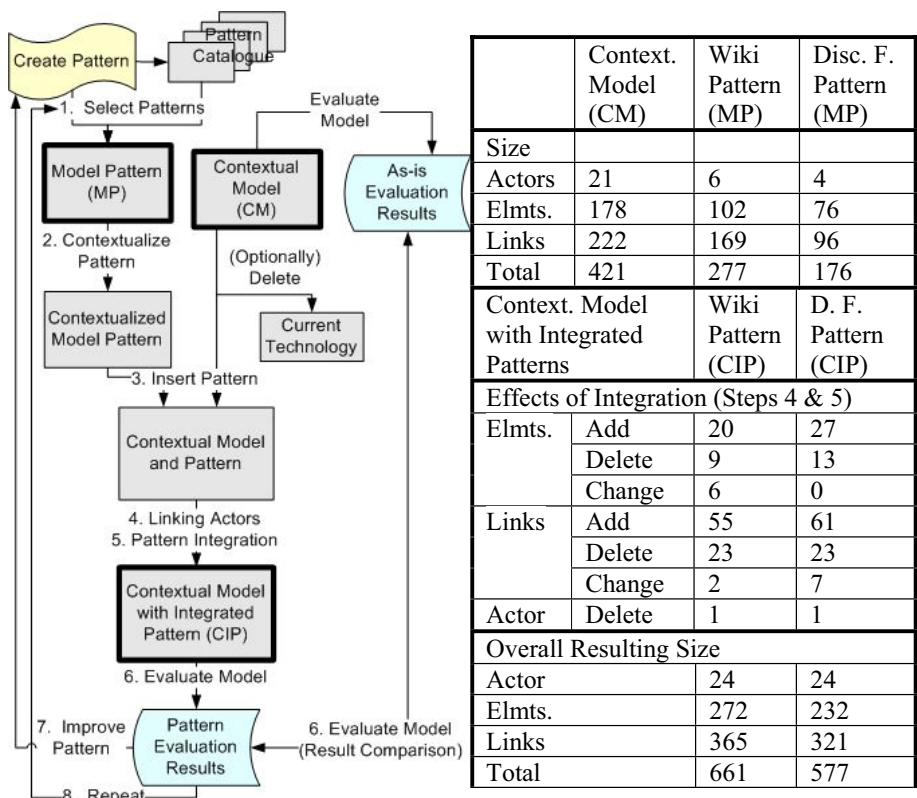


Fig. 2. Pattern Methodology and Resulting Measurements

configuration proved to be most successful, overall, based on an evaluation of the goals in the contextual model, the wiki technology did not seem to meet the needs of the organization. The discussion forum pattern, with differing features than the existing discussion forum, seemed to show more promise.

7. Improve Pattern & 8. Repeat. Use the experience of inserting and integrating the model patterns to make any necessary adjustments or improvements to the pattern. In this way, existing patterns can be gradually validated through iterative use and modification. For each relevant pattern, repeat steps 1 to 7.

The results of both of the studies we have conducted are analyzed in Section 7.

5 Exploratory Experiment: Classroom Setting

Although results collected in the case study have potential to address our research questions, the study had several limitations, several of which related to internal validity. First, the modelers who applied the model patterns were often their creators, which is not necessarily the case in pattern-oriented approaches. Second, the evaluation of the pattern approach in the case study was performed by the authors of

this work. In order to address some of these limitations, we designed and executed a follow-up experiment in order to find further evidence to address our research questions.

5.1 Exploratory Experiment Context and Preparation

The experiment took place in a graduate course of a school with a focus on business and technology. Thus, the students had a mixed background of technical and business experience. The students had some knowledge of the i* Framework through previous courses, but they had not applied it extensively and could be considered to be novice modelers. The study was introduced to the course as one of the course assignments; however, participation was anonymous and voluntary, not affecting grading. Six students opted to participate in the experiment.

The student assignment was divided into three parts. Part A (Contextual Model Creation) simulated the creation of a contextual model, with each student analyzing a type of information technology as applied to a collaborative work setting. All students analyzed the same work setting, but used different technologies. Models were evaluated to explore the effectiveness of the technology. Part B (Pattern Creation) involved the creation of a model pattern, with each student producing a pattern for a technology that they did not choose in the first stage. The third part, Part C (Pattern Integration), required the students to apply and contextualize a selected pattern produced by another student into their model created during Part A. Hints for integration were given to the students by describing some of the steps presented in the methodology in section 4.2. Questions were posed in the assignment to qualitatively assess effort (Q1), coverage (Q2) and complexity (Q3) of various steps in the assignment.

5.2 Qualitative Analysis of Experimental Results

Q1: To address modeling effort, the students were asked which of the assignment activities were the most difficult for them to complete. One student said this was the construction of the Part A (contextual) model, two students indicated that making the Part B pattern was either the most difficult or time consuming to construct, complaining about the difficulty of having to make a more abstract model, and two student said that the integration in Part C was the most time consuming task, with three students complaining about the difficulty of understanding the Part B model. The last student did not clearly pick a task as most difficult.

Q2: To address their perception of coverage, the students were asked the following question, with student answers summarized in Table 1 and the # symbol indicating answers that were missing or unclear.

How would you describe your confidence on the correctness (including accuracy and completeness of coverage) of the models and analysis results of:

1. The Part A model before you performed Part C?
2. The Part A model after you performed Part C?
3. The Part C Model? (in comparison to the Part A model)

Table 1. Summary of Student Answers for Question 2

	Q 2.1.	Q 2.2.	Q 2.3.
Student 1	Correctness: above average Coverage: not sufficient	Accuracy: good Completeness: not as good as part C	#
Student 2	Correctness: High confidence Coverage: cannot be determined	Correctness: High confidence Coverage: cannot be determined	C more complete in coverage
Student 3	Accuracy: good Completeness: good	Completeness: was not as good as thought	Part C model more complete
Student 4	Completeness and Accuracy: not confident (due to lack of i* experience)	Completeness and Accuracy: not confident (due to lack of i* experience) did add more things	Completeness and Accuracy: more confident, but still not completely confident
Student 5	Accuracy: high	Accuracy: not as high as thought	Part C models most accurate and correct
Student 6	Quality and Accuracy: Not confident	Lots of details left out in part A	#

Q3: When asked which models would be the easiest to understand for themselves or for others (related to our research question of model complexity), four students said that the Part C model is easiest for them to understand, while two indicated Part A. However, only two students clearly said that the Part C models would be easiest for others to understand, with one student indicating it would depend on the modeler's experience and another expressing concern about the complexity of Part C.

In addition, when asked about the quality of the model pattern produced by another student, received in Part C, five of the students complained about some aspect of the pattern they were supposed to apply, including completeness, ambiguity and complexity. However, four of these students, as well as the sixth student, listed positive aspects of applying patterns, including quality and knowledge previously missing.

Finally, despite the concerns expressed, when asked about their overall experience with using patterns in the assignment, five of them said they would use patterns again, although one indicated that only if the pattern was created by a reputable source. The validity of this and other evidence collected is discussed in Section 8.

6 Interpretation and Discussion of Results

In this section, we interpret and discuss the collected evidence from our studies in the light of our three driving research questions.

Q1: Do model patterns help reduce modeling effort? Assuming that patterns are readily available (leaving costs related to pattern construction aside), this research question can be affirmed when the integration of model patterns is less costly than the development of the corresponding parts of non-pattern models.

Case Study: We can acquire an *estimation* of modeling effort by examining the size of this section of the model its sub-agents and related dependencies. By comparing these measures to the amount of effort put into the integration of the discussion forum pattern into the same model, we derive evidence with respect to the question at hand. In our study, the contextually developed discussion forum model has 42 elements and 52 links, compared to the integration of the discussion forum model pattern which required the modification of 40 (13 deletions and 27 additions) elements and 84 links (23 deletions, 61 additions and 7 modifications).

Experiment: To make the experiment's results comparable to our case study results, we would ignore the effort put into the Pattern Creation of the assignment, despite several complaints about the difficulty of this activity. However, apart from pattern construction, there is the act of understanding the pattern sufficiently in order to apply it. As reported, five of the students expressed concerns about their ability to understand the incoming patterns. Furthermore, only one student indicated that Contextual Model Construction was the most difficult to construct while at least three students indicated difficulties with Pattern Integration.

Combined: Examining the Case Study evidence, it appears that the integration of the discussion forum model pattern required at least as much effort as modeling the technology within the contextual model. Considering the experiment, it seems that in addition to problems understanding the incoming pattern, the integration of a model pattern into a contextual model was generally thought to be more difficult than creating the contextualized model. These results are in clear conflict to our predictions, and especially surprising as we have already left the costs related to pattern construction out of the equation. The difference in the level of effort required to integrate patterns between studies may indicate that effort depends heavily on experience, as the case study was performed by experienced i* modelers, while the students in the experiment were i* novices.

Q2: Do model patterns help increase model coverage? This research question can be affirmed when the application of patterns leads to models that cover more *relevant* aspects of the domain than non-pattern models.

Case Study: We have found that patterns have a significant impact in this regard: by replacing the contextually developed discussion forum model with a discussion forum model pattern, model coverage increased along several dimensions: the integration of the pattern introduced 10 additional goals (+143%), 43 additional softgoals (+96%), 49 additional "help" contribution links (+87%) and 14 additional means-ends relationships (+1400%). We can surmise that these were additions of relevant constructs as irrelevant model sections were removed during the contextualization of the model pattern.

Experiment: Even though the students did not have high confidence in the coverage of their contextual models before pattern integration, three students indicated that the integrated models were the most complete and at least two students noticed detail left out of the contextual model after completing the Pattern integration.

Combined: The evidence found in both studies therefore suggests that the adoption of model patterns can have a *positive influence* on *elaborating* i* models with respect to model coverage.

Q3: Do model patterns help decrease complexity? This research question can be affirmed when the application of patterns leads to models which are significantly less complex than non-pattern models.

Case Study: In our case study, all developed model patterns were significantly *more complex* (i.e. contained more elements and links) than their technology counterparts in the contextual models, as discussed in the Q2 analysis. In the example where we have introduced the discussion forum model pattern to replace its contextual counterpart, an overall increase of modeling elements and links of 30% and 43%, respectively, could be observed. Results for replacing the contextual discussion forum model with the wiki pattern showed similar trends. However, the use of patterns can be said to modularize the model development process, and, as the patterns are significantly smaller than the contextual models before and after integration, the complexity of any steps performed with *only* the patterns would be simpler than working with the larger contextual model.

Experiment: We can examine questions relating to the quality of the model pattern and ease of comprehension as measures of model complexity. As mentioned, five of the students expressed concerns about their ability to understand the incoming patterns. In addition, although four of the students indicated that the integrated model would be the easiest to understand, there was concern over the ability of other to understand these models.

Combined: Despite the possible benefits of modularization, as well as the student's purported ability to understand their own integrated models, we are led to doubt an overall reduction in complexity from the use of model patterns. In fact, measuring complexity from model size, the case study results indicate that pattern application may actually increase model complexity.

7 Threats to Validity

Construct Validity: The constructs we intended to investigate in our study were *effort*, *model coverage* and *model complexity*. In our case study we measured the *effort* involved in model construction by measuring the amount of necessary model changes (additions, deletions). In doing that, we aimed to eliminate confounding factors such as the varying skills of modelers with a particular modeling tool. However, our approach does not mitigate the potential influence of varying cognitive efforts. In fact, our observations indicate that the act of integrating a pattern into a model may require more cognitive effort than the creation of corresponding, contextual models, which represents an interesting finding.

In the KHP study, we measured *model coverage* by investigating whether the total amount of modeling elements and links increased or decreased after integration of the model patterns into the contextual model. These changes were made with the relevance of these elements in mind. A potential threat to validity is the subjective nature of "relevance" in general. We tried to mitigate this factor by involving a modeler that has a good understanding of the case study organization. Our case study used the size of the models, including elements and relations, as a measure of *model complexity*. We argue that this represents a suitable surrogate measure for an

exploratory case study. To address issues with the means of measuring effort, coverage and complexity in our case study, our exploratory experiment instead used a qualitative judgment of these aspects as reported by the student participants.

Internal Validity: The internal validity problems of the case study were discussed in Section 6. In the experiment, pattern creation was performed by novice modelers, whereas in pattern theory, patterns are typically developed by experts in the domain and pattern creation. We attempted to mitigate these effects by providing resources on the technology subject matter of the patterns and by providing sufficient i* training. However, results may have differed if the patterns were created by more experienced individuals.

External Validity: Because the experiment and case study were performed using the i* Framework, it is difficult to generalize findings to other modeling frameworks. However, several of our findings may generalize to other agent-oriented, goal modeling frameworks, such as the fact that pattern integration involves significant effort or that patterns have the potential to increase model coverage.

As always, there are external validity issues with the use of students as research subjects, especially when the sample size is small. However, this particular group of students represented a fairly diverse background, having a mixture of academic and business experiences. Furthermore, the subjects had a novice level of expertise in use of the i* Framework, making it difficult to generalize to more experienced modelers. In contrast, the modelers in the case study were experienced with the i* Framework.

Both the case study modelers and the students were experts in their respective domains, KHP and a collaborative work setting. It is possible that differing levels of expertise may produce different findings. However, the issue of expertise in the pattern technology may be yet more relevant, with participants in both studies having varying levels of expertise in the technologies modeled.

The differences in the contexts of our investigations increase our confidence that the results would generalize to other settings. However, it is still possible that some domains may be more amenable to pattern application than others.

The results of our study may depend on the nature of the patterns we use. Employing a variety of pattern creators in both the case study and experiment increases our confidence that the results would generalize to different sized and scoped patterns, but i* patterns defined in a different way may produce different results.

Reliability: Making the methodology we followed explicit increases our confidence that our findings can be reproduced by others. Other than the small number of participants, there is nothing to indicate that, given similar settings, both of our studies would not produce similar results.

8 Related Work

In i* modeling, patterns have not been applied extensively, but some reports are available. [15] and [16], for example, use the i* Framework to 1) construct agent-oriented strategic dependency patterns of different types of organizational structures and to 2) (re)construct traditional, object-oriented patterns in an agent-oriented

fashion. In addition, [18] uses i* strategic dependency and strategic rationale diagrams to capture and encapsulate knowledge about possible design trade-offs of submarine maneuvering systems for reuse in future engineering efforts. Reusable security patterns, expressed in the i* Framework, are introduced in [17]. While these examples demonstrate the potential of patterns for agent-based, social focused modeling, testing the assumption that a pattern-based approach actually improves modeling was not in the focus of these investigations.

In the broader context of *requirements engineering*, patterns have been proposed and used for many different purposes. Patterns were proposed and investigated as a means for organizing and documenting, for example, functional and non-functional requirements knowledge ([5], [9]) and for capturing knowledge about requirements engineering techniques and strategies [11]. Examples include patterns for refining requirements [6], and dealing with conflicts [19]. Beyond these approaches, patterns were suggested to act as solution templates for requirements specification (Hosoya in [10]), and as guidelines for performing and improving the requirements process [12]. Finally, patterns were investigated as references for assuring the quality of specifications (Hanyuda in [10]). In *software engineering* in general, patterns have a longer tradition. Beyond the influential work on object-oriented patterns (including [3] and [4]), a series of approaches for utilizing *agent-oriented* design patterns have been proposed including [2] and [14].

9 Conclusions

Execution of the studies in this work has revealed some limitations to the use of model patterns in i*. For instance, contrary to our expectations, replacing the technologies in our case study with the two patterns did not have a large effect on the overall goals of KHP's actors. This emphasizes that the application of patterns to a model is a bottom-up (solution driven) approach, whereas the traditional goal modeling approach is predominately top-down (goal driven). Although applying patterns was useful for improving coverage, further brainstorming is required to sufficiently satisfy the goals of the organization.

Execution of the experiment revealed potential difficulties with the construction and comprehension of model patterns. Some students had difficulty constructing patterns capturing abstract situations. Patterns created by other students were often difficult for a student to understand or apply. Further studies should test whether these issues are as apparent when models are created and used by experienced modelers.

Can patterns improve i* modeling?

Q1-Effort: We have found that several assumptions of pattern theory seem to be questionable when applied to i* modeling. Even when we took the effort necessary for pattern creation out of the equation, we found empirical evidence that suggests that patterns increase modeling effort for novice users, and do not decrease effort for more experienced users.

Q2-Coverage: The findings of our exploratory investigations suggest that the utilization of patterns can address issues identified with i* modeling related to coverage by integrating broader domain knowledge.

Q3-Complexity: We could not find evidence that patterns help in reducing complexity in an i^* context. In fact, our quantitative case study findings suggest the opposite: pattern integration almost always led to an increase of modeling elements. Our qualitative experimental findings also point to an increase in the complexity of models containing patterns, especially for those not creating the models. However, because patterns also modularize the domain and can be inspected independent from their contexts, patterns might nevertheless support analysts in dealing with large-scale models. Further studies should investigate this possibility.

Combining these preliminary observations, we can make the assertion that the decision to apply patterns in a given situation can be made based on certain factors including the importance of model coverage and the experience of the modelers. If model coverage, including related factors such as accuracy and correctness, are strongly desired, applying a tested and reputable pattern can be beneficial, especially if being applied by experienced modelers. However, if reduced effort and complexity are favored over coverage, or if modelers are inexperienced, a pattern approach may be less appropriate.

In this paper, we have investigated the application of model patterns in the presence of existing contextual models. One promising further application is the utilization of patterns at the beginning of the modeling process, where contextual models have not yet been created. In this situation, model patterns could be used as seeding elements for the construction of contextual models, eliminating the effort of pattern integration. Relevant topics for future research brought to light by our exploratory studies include examining the impact of patterns on model comprehension and correctness, as well as further investigating the effect of modeler experience and domain expertise on the ability to effectively apply patterns.

Acknowledgements

We thank the experiment participants as well as all the management and staff at Kids Help Phone for allowing us to conduct these studies. Funding for this work was provided by Bell University Labs (BUL), the National Sciences and Engineering Research Council of Canada, the Ontario Graduate Scholarships Program, the Austrian COMET Program - Competence Centers for Excellent Technologies, and the FWF Austrian Science Fund.

References

1. Alexander, C.: *The Timeless Way of Building*. Oxford Press, Oxford (1979)
2. Aridor, Y., Lange, D.B.: Agent design patterns: Elements of agent application design. In: 2nd Int. Conference on Autonomous Agents, pp. 108–115. ACM Press, New York (1998)
3. Beck, K., Coplien, J.O., Crocker, R., Dominick, L., Meszaros, G., Paulisch, F., Vlissides, J.: Industrial Experience with Design Patterns. In: 18th International Conference on Software Engineering (ICSE), pp. 103–114. IEEE Press, New York (1996)
4. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: *Pattern-Oriented Software Architecture. A System of Patterns*, vol. 1. Wiley, New York (1996)

5. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer, Norwell (2000)
6. Darimont, R., van Lamsweerde, A.: Formal refinement patterns for goal-driven requirements elaboration. *ACM SIGSOFT Soft. Eng. Notes* 21(6), 179–190 (1996)
7. Easterbrook, S.M., Yu, E., Aranda, J., Fan, Y., Horkoff, J., Leica, M., Qadir, R.A.: Do Viewpoints Lead to Better Conceptual Models? An Exploratory Case Study. In: 13th IEEE Int. Conference on Requirements Engineering (RE 2005), pp. 199–208. IEEE Press, New York (2005)
8. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns - Elements of Reusable Object-Oriented Software. Addison Wesley, Reading (1995)
9. Gross, D., Yu, E.: From Non-Functional Requirements to Design through Patterns. *Req. Eng.* 6(1), 18–36 (2001)
10. Hagge, L., Houdek, F., Paech, B.: Workshop Summary of the International Workshop on Requirements Engineering Patterns. In: International Workshop on Requirements Engineering Patterns, In conjunction with RE 2004 (2004), <http://rep04.desy.de/>
11. Hagge, L., Lappe, K.: Sharing requirements engineering experience using patterns. *IEEE Software* 22(1), 24–31 (2005)
12. Hagge, L., Lappe, K.: Using Requirements Engineering (RE) Patterns for Organizational Learning. *Journal of Universal Knowledge Management* 1(2), 137–148 (2006)
13. Horkoff, J., Yu, E., Liu, L.: Analyzing Trust in Technology Strategies. In: Int. Conference on Privacy, Security and Trust (PST 2006), pp. 21–32. McGraw-Hill, New York (2006)
14. Kendall, E.A., Krishna, P.V.M., Pathak, C.V., Suresh, C.B.: Patterns of intelligent and mobile agents. In: Second International Conference on Autonomous Agents, pp. 92–99. ACM Press, New York (1998)
15. Kolp, M., Giorgini, P., Mylopoulos, J.: Organizational Patterns for Early Requirements Analysis. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, p. 1030. Springer, Heidelberg (2003)
16. Kolp, M., Giorgini, P., Mylopoulos, J.: A Goal-Based Organizational Perspective on Multi-Agent Architectures. In: Meyer, J.-J.C., Tambe, M. (eds.) ATAL 2001. LNCS (LNAI), vol. 2333, pp. 128–140. Springer, Heidelberg (2002)
17. Mouratidis, H., Giorgini, P., Schumacher, M.: Security Patterns for Agent Systems. In: 8th European Conference on Pattern Languages of Programs (EuroPLoP), pp. 25–29. Wiley, New York (2003)
18. Pavan, P., Maiden, N.A.M., Zhu, X.: Towards a Systems Engineering Pattern Language: Applying i* to Model Requirements-Architecture Patterns. In: 2nd Int. Workshop from Software Requirements to Architectures (STRAW 2003), co-located with ICSE (2003)
19. van Lamsweerde, A., Darimont, R., Letier, E.: Managing conflicts in goal-driven requirements engineering. *IEEE Transactions on Software Eng.* 24(11), 908–926 (1998)
20. Yu, E.: Modelling Strategic Relationships for Process Reengineering. PhD Thesis, Department of Computer Science, University of Toronto, Toronto, Canada (1995)
21. Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In: 3rd IEEE Int. Symp. on Requirements Engineering (RE 1997), pp. 226–235. IEEE Press, New York (1997)

Discovering Web Services to Improve Requirements Specifications: Does It Help?

Konstantinos Zachos, Neil Maiden, and Rhydian Howells-Morris

Centre for HCI Design, City University, London
kzachos@soi.city.ac.uk, n.a.m.maiden@city.ac.uk,
rhydianmorris2002@hotmail.com

Abstract. Service-centric systems pose new opportunities for when engineering requirements. This paper reports an evaluation of software tools with which to exploit discovered services to improve the completeness of requirements specifications. Although these tools had been evaluated previously in facilitated industrial workshops, industrial users had not used the tools directly. In this paper we report 2 industrial uses and evaluations in which experienced analysts used the tools directly on 2 real-world requirements projects. Results reveal that analysts used the tools to retrieve web services that could implement specified requirements, but analysts were less able to improve these requirements in light of the retrieved services. Results have implications for iterative service discovery processes and service discovery algorithms.

1 Developing with Web Services

Web and software services are operations that users access via the internet through a well-defined interface independent of where the service is executed [1]. Service-centric systems integrate software services from different providers into applications that discover, compose and monitor these services. Developments in service-centric computing have been rapid [2], but there has been little reported research to address how to engineer service-centric systems.

As we have reported previously [3], one consequence of service-centric systems is that requirements processes might change due to the availability of services. Discovering candidate services can enable analysts to increase the completeness of system requirements based on available service features. We have researched new tools and techniques to form service queries from incomplete requirements specifications as part of the EU-funded SeCSE Integrated Project. Although the effectiveness of these tools to increase requirements completeness was demonstrated in workshops, in which stakeholders worked with the tools through facilitators and scribes [4], we still lacked empirical evidence of whether analysts can use and benefit from these tools directly. Therefore we made the tools available for use by SeCSE's industrial partners on service-centric systems development projects. This paper reports results from the requirements phases of projects at 2 of these partners – a large multi-national consultancy and a small software house providing applications. Results were used to investigate 2 research questions about the usefulness of the SeCSE tools:

- Q1.** Can the tools retrieve specifications of services compliant with requirements specified by analysts in service queries?
- Q2.** Can analysts make requirement specifications more complete using the retrieved service specifications?

To answer these 2 questions we collected data about the requirements specified by analysts, the service specifications retrieved using service queries composed of these requirements, analysts' decisions to retain or reject each of these services, changes to requirements in light of service specifications, and qualitative statements made by analysts. Q1 was answered using analyst decisions to retain or reject each retrieved service. Q2 was answered using post-retrieval changes that analysts make to use case and requirement specifications.

Sections 2 and 3 of this paper describe SeCSE's service-centric requirements process and tools. Section 4 introduces the 2 industrial users of these tools and the evaluation method, and sections 5 and 6 report results from the 2 evaluations. Section 7 answers the 2 research questions and reports some threats to validity. The paper ends with future work on new software modules to support the specification of requirements from retrieved web services.

2 Discovering Services in SeCSE

In previous SeCSE work we had reported an iterative and incremental requirements process for service-centric systems [5]. Requirements analysts form service queries from a requirements specification to retrieve services that are related to the requirements. Descriptions of these retrieved services are explained to stakeholders, then used to refine and complete the requirements specification to enable more accurate service retrieval, and so on.

Relevance feedback, as it is known, has important advantages for the requirements process. Stakeholders will rarely express complete requirements at the correct levels of abstraction and granularity to match to the descriptions of available services. Relevance feedback enables service consumers and analysts to specify new requirements and re-express current ones to increase the likelihood of discovering compliant services. Furthermore accurate relevance feedback provides information about whether requirements can be satisfied by available services, to guide the analysts to consider build, buy or lease alternatives or to trade-off whether requirements can be met by the available services.

The process has 2 important features. Firstly, to ensure its industrial uptake, the process uses established specification techniques based on structured natural language. For example, to specify system behaviour the process supports UML use case specifications. To specify the required properties in a testable form for generating service monitoring policies it supports the VOLERE requirements shell [6]. As such the process extends the Rational Unified Process (RUP) without mandating unnecessary specification or service querying activities.

Secondly the process uses services that are discovered from service registries to challenge system boundaries and discover new requirements. For example, if no services are found with an initial query, SeCSE provides advice on how to broaden the

query to find services that, though not exactly matching the needs of the future system, might provide a useful basis for further specification.

To support the iterative and incremental requirements process we implemented the SeCSE service discovery environment. The next section describes this environment.

3 SeCSE's Service Discovery Environment

The environment has 4 modules: (i) service registries; (ii) UCARE, a module to document requirements and generate service queries; (iii) EDDiE, the service discovery engine, and; (iv) the Service Browser module for reviewing and selecting retrieved services. We describe these 4 modules in turn.

3.1 SeCSE's Service Registries

The environment discovers services from federated SeCSE service registries that store both the service implementation that applications invoke and one or more facets that specify different aspects of each service. Current service registries such as UDDI are inadequate for discovering services using criteria such as cost, quality of service and exception handling. Therefore SeCSE has defined 6 facets of a service – signature, description, operational semantics, exception, quality-of-service, cost/commerce, and testing [7] – that specify features that are important when discovering services. Each facet is described using an XML data structure. The environment uses the description and quality-of-service facets of each service. Figure 1 shows part of the service description facet of one service retrieved in 1 of the 2 reported evaluations. The quality-of-service facet is used to refine selection once services are discovered. SeCSE's service registries are implemented using eXist, an Open Source native XML database featuring index-based XQuery processing, automatic indexing.

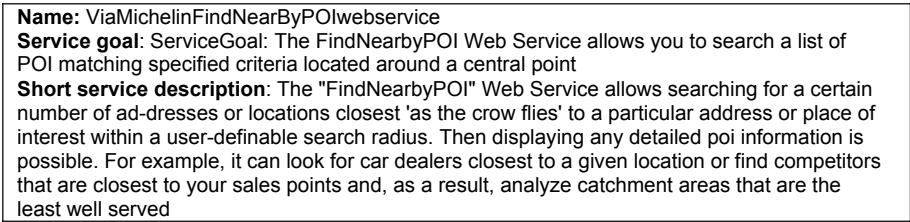


Fig. 1. Example of part of one service with SeCSE's description facet

3.2 The UCARE Requirements Module

Analysts express requirements for new applications using UCARE, a web-based .NET application. UCARE supports tight integration of use case and requirements specifications – a requirement expressed using VOLERE can describe a system-wide requirement, a requirement on the behavior specified in one use case, or a requirement on the behavior expressed in one use case action. UCARE allows analysts to create service queries directly from use case and requirements specifications.

At the start of the requirements process, analysts work with stakeholders to develop simple use case précis that describe the required behaviour of a new system. Figure 2(a) shows a use case précis expressed in UCaRE, taken from one of the reported evaluations, to specify how a user shall use an on-line ticket searching application. A second précis also used in the evaluation is reported in a readable form in Figure 3. Figure 2(b) shows a requirement, also from these partners, associated with the précis expressed using the UCaRE VOLERE shell. Larger versions of the screenshots in Figure 2 are available at [8]. Requirements also used in the evaluation are reported in readable form in Figure 3.

The analyst then uses the simple tick-box feature shown in Figure 2(c) to select attributes of use cases and requirements to include in a service query. Each service query is formed of one or more elements of a pre-defined type such as a requirement description or rationale, or a use case précis, pre-condition or action. UCaRE maps these element types to service query elements to deliver the seamless integration of service querying with requirements specification, as described at length in [3]. The

Figure 2 consists of three screenshots from the UCaRE interface. Screenshot (a) shows the 'Use Case Name' field with 'Estimate Expense' and the 'Actors' field with 'Consultant search', 'Consulting Manager', and 'Calculate D.'. The 'Precis' field contains a detailed description of the use case. Screenshot (b) shows the 'Requirement Template' dialog box with 'Requirement ID: AR1', 'Requirement Type: Availability', and 'Description: Must be available during local office hours'. Screenshot (c) shows the 'Functional Requirement(s)' section with a table of requirements and their sources, including 'CA development guidelines' and 'CA Corporate finance standards'.

Fig. 2. Specification of a use case (a) and requirement (b) in UCaRE, and selection of use case and requirements attributes to generate service queries (c)

Precis: All users can search events and tickets by use of search function. She can search event and ticket on base of event date or event name or event place or interpreter (band, sport team, etc.) name. She can specify also type of event. If search result is more then one, they are displayed like list. Number of list rows on page can be limited by application variable.

AR: Must be available during local office hours.

FR: For registered and logged user must be possibility to book or purchase tickets directly from list.

Fig. 3. A simple use case précis and requirements for the ticket searching application used in one of the industrial evaluations, which are used to formulate queries and discover services

analyst then refines each generated service query using the names and locations of registries to search, the maximum number of services to retrieve, and the parts of speech (e.g. noun, verb and adjective) in the service query text to search on.

An analyst using UCARE can generate one or more service queries from the specification of a system. Each query is a structured XML file containing structured natural language statements. Because these statements are derived from requirements and use cases, each is potentially ambiguous and incomplete. EDDiE, the service discovery engine, was designed to handle this ambiguity and incompleteness.

3.3 The EDDiE Module

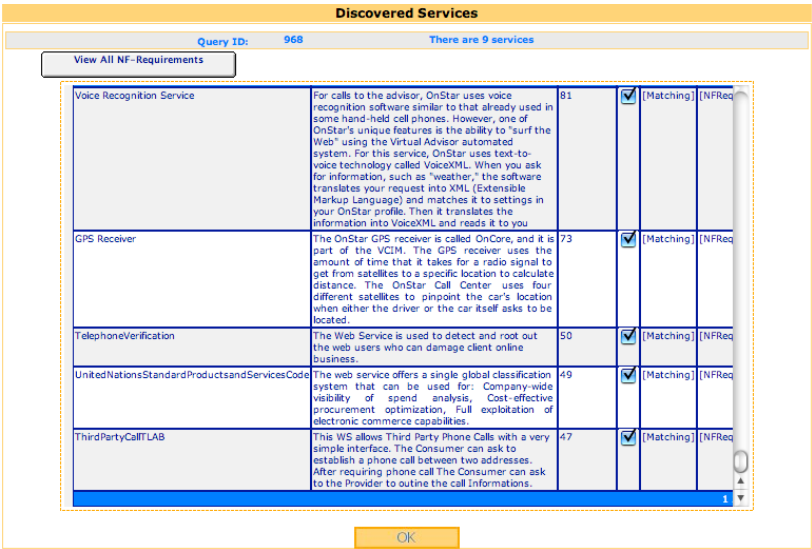
The purpose of EDDiE is to discover descriptions of candidate services using the service description facet with queries composed of information such as that in Figures 2 & 3. Other requirement types and service facets such as quality-of-service and cost fulfil important roles during service selection once discovered using the Service Browser module.

The EDDiE algorithm has the 4 key components. In the first the service query is divided into sentences, then tokenized and part-of-speech tagged and modified to include each term's morphological root (e.g. *displayed* to *display*, and *tickets* to *ticket*). Secondly, the algorithm applies procedures to disambiguate each term by defining its correct sense and tagging it with that sense defined in the WordNet online lexicon [9] (e.g. defining a *ticket* to be a *a commercial document showing that the holder is entitled to something (as to ride on public transportation or to enter a public entertainment rather than a list of candidates nominated by a political party to run for election to public offices)*). Thirdly, the algorithm expands each term with other terms that have similar meaning according to the tagged sense using WordNet, to increase the likelihood of a match with a service description (e.g. the term *ticket* is synonymous with the term *order* or *voucher* which is also included in the query). This query expansion enables the algorithm to retrieve service specifications for service queries that share no common terms. In the fourth component the algorithm matches all expanded and sense-tagged query terms to a similar set of terms that describe each candidate service, expressed using the service description facet, in the SeCSE service registry. Query matching is in 2 steps: (i) XQuery text-searching functions to discover an initial set of services descriptions that satisfy global search constraints; (ii) traditional vector-space model information retrieval, enhanced with WordNet, to further refine and assess the quality of the candidate service set. This two-step approach overcomes XQuery's limited text-based search capabilities. The algorithm returns a set of retrieved service specifications and match scores ranked according to the semantic distance to the service query.

The EDDiE algorithm is described at length in [3].

3.4 The Service Browser Module

The Service Browser presents retrieved services to analysts and stakeholders. Services that attain a minimum threshold of match value are presented in a ranked order. The analyst can view all properties of the service description facet and corresponding use



The screenshot shows a window titled "Discovered Services". At the top, it says "Query ID: 968" and "There are 9 services". Below this is a button labeled "View All NF-Requirements". The main content is a table with five rows of services. Each row contains the service name, a detailed description, a match score, a checkbox, and a status label. The services listed are Voice Recognition Service, GPS Receiver, TelephoneVerification, UnitedNationsStandardProductsandServicesCode, and ThirdPartyCallTLAB. All services have a match score of 81, 73, 50, 49, and 47 respectively, and all are marked as "[Matching] [NFRReq]".

Service Name	Description	Match Score	Match Status	Requirement Status
Voice Recognition Service	For calls to the advisor, OnStar uses voice recognition software similar to that already used in some hand-held cell phones. However, one of OnStar's unique features is the ability to "surf the Web" using the Virtual Advisor automated system. For this service, OnStar uses text-to-voice technology called VoiceXML. When you ask for information, such as "weather," the software translates your request into XML (Extensible Markup Language) and matches it to settings in your OnStar profile. Then it translates the information into VoiceXML and reads it to you.	81	<input checked="" type="checkbox"/> [Matching]	[NFRReq]
GPS Receiver	The OnStar GPS receiver is called OnCore, and it is part of the VCM. The GPS receiver uses the amount of time that it takes for a radio signal to get from satellites to a specific location to calculate distance. The OnStar Call Center uses four different satellites to pinpoint the car's location when either the driver or the car itself asks to be located.	73	<input checked="" type="checkbox"/> [Matching]	[NFRReq]
TelephoneVerification	The Web Service is used to detect and root out the web users who can damage client online business.	50	<input checked="" type="checkbox"/> [Matching]	[NFRReq]
UnitedNationsStandardProductsandServicesCode	The web service offers a single global classification system that can be used for: Company-wide visibility of spend analysis, Cost-effective procurement optimization, Full exploitation of electronic commerce capabilities.	49	<input checked="" type="checkbox"/> [Matching]	[NFRReq]
ThirdPartyCallTLAB	This WS allows Third Party Phone Calls with a very simple interface. The Consumer can ask to establish a phone call between two addresses. After requiring phone call The Consumer can ask to the Provider to outline the call informations.	47	<input checked="" type="checkbox"/> [Matching]	[NFRReq]

Fig. 4. The Service Browser module, showing retrieved services, their specifications and match scores

case and requirement properties to enable understanding and selection. The analyst can also filter the services according to compliance or otherwise with non-functional requirements included in the service query. A screenshot showing candidate services retrieved for queries in one evaluation is depicted in Figure 4. A larger version of the screenshot in Figure 4 is available at [10].

Previously SeCSE’s service discovery environment had been tested with industrial users for its usability and core functionality [11]. It was also evaluated successfully in a half-day workshop at FIAT’s research centre to discover requirements for new automotive applications [4]. However, the FIAT analysts did not use the tool directly. In the remainder of this paper we report the next phase of evaluation, in which analysts used the environment on their own to specify requirements and retrieve candidate service specifications.

4 The Industrial Users and Evaluation Method

The 2 industrial users – both members of the SeCSE project – were CA and KD Software. CA is one of the world's largest IT management software providers. It undertakes core systems integration roles for clients seeking secure, service-oriented architectures. KD Software is an independent software developer in the Czech Republic that develops business systems using service-centric techniques.

Two experienced analysts – 1 from the UK office of CA and 1 from the Czech office of KD Software – undertook the evaluations. Both had extensive analytic experience and were familiar with UML. Both worked remotely at their offices, accessing SeCSE tools on servers based in London using thin web clients through their

organizations' firewalls. The CA analyst received initial on-site training with the tools whereas the KD Software analyst learned to use the tools independently using SeCSE user guides. The SeCSE tools searched 4 federated service registries located in Italy and Spain. They contained 154 service specifications for applications including weather reporting, flight booking and route planning taken from existing public UDDI registries and generated by service providers in SeCSE. Each service description was written by the original service provider and not modified prior to use.

The CA analyst specified requirements for a travel cost estimation application for CA consultants to use. The KD Software specified requirements for an outline ticket searching application. The analysts worked independently of each other, but undertook the same 6-step evaluation method. Each step is described in turn:

1. The application was analyzed using UML use case diagrams to generate use case specifications of the required behaviour of the application;
2. One or more use case specifications for use cases described in the diagrams were entered directly by each analyst into UCARE, and requirements were entered using the VOLERE requirements shell;
3. Each analyst used UCARE functions to generate one or more service queries per use case specification defined in UCARE during the second step;
4. Each analyst used EDDiE to retrieve services from SeCSE service registries using the service queries generated in the previous step. KD Software had specified and published one web service, called *KDTicketDataDelivery2a*, in the service registries which the analyst aimed to discover in the evaluation. CA had not published any service specifications, hence the evaluation investigated whether an uncontrolled set of available services could be used to support CA's requirements process;
5. Each analyst used the Service Browser to understand the service specifications retrieved from the registries and select those relevant to the generated queries;
6. Each analyst experimented with changes to use case and requirements specifications in light of the discovered services, documenting changes in use case and requirements specified in UCARE. Each analyst could then repeat steps 3-6 again until the evaluation was complete.

Each analyst undertook all 6 steps. The 6 steps provide reference steps during the descriptions of the 2 evaluations reported in the next 2 sections.

5 Results from the CA Evaluation

The CA travel cost estimation application was specified to support its consultants who travel to client sites then bill their time and expenses. The use case diagram for the application contained 15 use cases and 3 actors. The primary actor in the model was the consulting manager, who seeks to achieve goals such as create draft description of work, establish price, agree project terms and conditions, search for consultants, and review projects.

Several of the use cases in the diagram were expanded into use case specifications entered into UCARE. One such use case specification, *Estimate expense*, is shown in Figure 5. The problem statement outlined the existing problem. The précis described the required behaviour using unstructured text. The author also specified 4

Use Case ID	Estimate Expense
Actors	Consultant search, Consulting Manager
Problem statement	As part of the Consultant search function the Consulting manager needs to Estimate expenses. For this to be as accurate as possible the consulting manager has to get details of distance, travel mode and accommodation requirements costs. All of these costs.
Precis	When the consulting manager is matching consultant to company he needs to establish accurate costs to enable him to negotiate and agree a job price with the customer. Estimate expense works on distance of skilled consultant from the customer, what mode of travel is being used, what the contract says about travel expenses and what accommodation on site is required.
Functional Requirements	Response time should be < 1 minute.
Non-Functional Requirements	All costs to be calculated in US Dollars but values to be displayed in local currency based on that days interbank rate. Must be available during local office hours. Travel type must be shortest time of travel for consultant unless that gives a disproportionate cost increase.
Added Value	Currently done manually with online mapping and some rule of thumb measures.
Justification	Accurate estimate generate confidence and correct pricing of jobs
Triggering event	Sales notification of a contract negotiation that includes consultancy costs.
Preconditions	Draft description of work is available. DOW has been reviewed by the customer and agreed in principle.
Assumptions	Consultants are appropriately skilled for the work. Choice of travel mode is at the consultants discretion, discounted travel is not appropriate for the work.
Normal Course	1. Draft description of work review complete.
	2. Distance between consultant and customer is calculated.
	3. Cost of daily travel is estimated.
	4. Mode of transport is decided.
	5. Accommodation is accepted or rejected.
	6. Total cost calculated.

Fig. 5. The *Estimate expense* use case specification from the CA application

requirements – 1 functional and 3 non-functional – on the behaviour specified in the use case. A post-study review revealed that the specified functional requirement should be a performance requirement and at least one of the non-functional requirements can be interpreted as a functional one, so service discovery took place using incorrectly typed requirements. The use case normal course was composed of 6 actions that describe the required behaviour of the new expense estimating application.

During step 2 of the process the CA analyst commented that UCaRE was robust but necessitated the user guide to specify use cases. He also commented that there was no discernible difference in tool performance between access from the server site and remotely at CA offices, but remote access suffered from some Internet lag.

During step 3 the CA analyst successfully generated service requests and queries from the use case specification. The service query retrieved 14 candidate service specifications from the registries containing the 154 service specifications. Table 1 lists the names of the retrieved services in match order and the CA analyst's decisions to retain or reject each service using the Service Browser module.

During step 5 of the process the CA analyst retained 6 and rejected 7 of the 14 retrieved services to invoke in the future travel cost estimation application. An 8th was also rejected but identified as potentially useful to a related CA application. Over half of the services retrieved by EDDiE were deemed to be incorrect by the analyst for the specified service query.

Table 1. Ranked services retrieved by EDDiE for the CA *Estimate expense* use case specification, and the decision to retain or reject each of the services

Discovered Service Name	Decision to retain or reject service
Weblogwebservice	- Rejected -
AdressMeister	+ Retained +
Webservice search	- Rejected -
XigniteCurrencies	+ Retained +
FoldCalc	- Rejected -
XgniteEdgar	Rejected, but could be useful in another application
QuoteAndMarketData	- Rejected -
ThirdPartyCallTLAB	- Rejected -
SendSMSTLAB	+ Retained +
Mobile7NavigationKit	+ Retained +
CreditCardVerifier	- Rejected -
XnavigationCEFRIEL	+ Retained +
TimeServiceCEFRIEL	+ Retained +
EmailVerifier	- Rejected -
XigniteDataSet	- Rejected -

There was no relationship between the services retained and the EDDiE ranking of these services. Short descriptions of the 6 retained services and the reasons for retaining them are reported in Table 2.

Table 2. Retained services and the CA analyst's rationale for their retention as relevant to the CA application

Service Name	Service Description	Rationale for retention
AdressMeister	Address Meister is a web-service for postal address verification and correction. It provides current, high-quality address data and verification logic without the cost and complexity of maintaining the nation's address database in-house. The service can be used by e-businesses to verify the addresses provided to them on their websites.	Verifying if the address is correct of both consultant and customer.
XigniteCurrencies	This web service provides real-time currency data (foreign exchange rate) for more than 170 currencies. Convert the US dollar amount to other currencies using real-time currency exchange rates returned by this currency web service	All internal currencies in USD, therefore currency conversion is required
SendSMSTLAB	This WS allows sending and monitoring SMS with a very simple interface. The Consumer can ask to send an SMS text to a list of addresses through the GSM network. After requiring SMS sending The Consumer can ask to the Provider to outline the delivery status of his request.	Communication of authorisation to consultant
Mobile7NavigationKit	ROUTE 66 Mobile 7 determines its position using an advanced high sensitive wireless GPS receiver, guiding the user with turn-by-turn voice instructions and on-screen directions to its destination. A new navigation display has been developed providing users with all vital travel information on a single clear screen of their smartphone including 3D map display, turn arrows and navigation guidance, as well as the ability to dial points of interest directly from the map.	Used to calculate distance between consultant and customer locations
XnavigationCEFRIEL	Especially useful for car drivers. You may want to know the duration in time of your trip, given the geographical positions of the departure and arrival places.	Required for estimating journey costs/time
TimeServiceCEFRIEL	This service computes the difference in time of two given moments.	Used in many places, for delivering time differences, e.g how long has the negotiation been progressing

The reasons given by the CA analyst demonstrate the usefulness of the retrieved services for estimating expenses. The *AdressMeister* service could be invoked to verify client addresses, the *XigniteCurrencies* service was needed to compute costs in a single currency, invoking the *SendSMSSTLAB* service would deliver a means of communicating system outputs to the consultant, the *Mobile7NavigationKit* service would calculate distances between locations deemed pivotal to the application, the *XnavigationCEFRIEL* service could be invoked to compute the travel time – also pivotal to the application, and the *TimeServiceCEFRIEL* service would provide important timing data to the application.

During step 6 of the process the CA analyst used the specifications of the 6 retrieved services to generate 2 new requirements and improve a 3rd one in UCaRE. Table 3 lists the 2 new requirements and changed third one, along with the rationale for these requirements provided by the CA analyst.

Table 3. New and changed requirements generated from relevant feedback from the 6 retained services retained within the Service Browser

Type of Edit	Requirement Description	Rationale
New requirement	Web based access for the remote use by consulting manager	Consulting manager may have to authorise travel when travelling themselves
New requirement	The application must present 2 alternative methods of travel, Lowest cost and shortest time	Lowest cost requirement is not the only requirement, see changed requirement below
Changed requirement	Travel type modified so that the travel is short only if it is cost effective	Removed requirement for lowest cost travel.

After the evaluation we examined the granularity of the specified requirements and retrieved web services. Both new and original requirements were coarser grain than retrieved web services that implemented atomic functions that, if invoked, were insufficient on their own to satisfy a requirement. Therefore EDDiE was able to retrieve web services that were finer-grain than the requirements in the service queries, but this difference in granularity might have impacted on further requirements generation.

Using the revised use case and requirement specifications the CA analyst returned to step 3 of the process and generated a new service query. EDDiE returned the same 14 service specifications in the same order, although some of the MatchValues were slightly different to the MatchValues returned for the original service query. Because of this second result, the CA analyst concluded the evaluation. After the evaluation he reported that he was unable to use the Service Browser to review the service specifications and their similarities with requirement and use case attributes in the service query effectively, and this made service selection activities difficult.

6 Results from the KD Software Evaluation

During steps 1 and 2 the KD analyst used the SeCSE tools to specify 10 use cases and the associated requirements on the ticketing selection application, then discover services from registries (steps 3 & 4) that were then browsed, selected (step 5) and used to revise the use cases and requirements (step 6). One of these use case specifications is reported in Figure 6. The specification reveals evidence that the KD analyst had also attributed the wrong type to some of the requirements used to generate service queries.

Use Case ID	KD UC Ticket Searching
Actors	User (Unregistered User, Registered User, Administrator)
Problem statement	Providing events and ticket information search
Precis	All users can search events and tickets by use of search function. She can search event and ticket on base of event date or event name or event place or interpreter (band, sport team, etc.) name. She can specify also type of event. If search result is more then one, they are displayed like list. Number of list rows on page can be limited by application variable.
Functional Requirements	FR: Application UI must be standard internet browser. FR: Ticket data are provided by any Ticket data delivery service. FR: Ticket data can be searched in own or third party database. FR: Tickets or events can be searched according to several criteria (ticket database connection, event name, date, place, event type, etc.). FR: The search results must be in list form with possibility to limit size (rows number).
Non-Functional Requirements	NFR: Availability 99,9% NFR: Delay max.30 second
Added Value	Searching functionality is open, it can works with several ticket databases.
Justification	User needs information for ticket purchase.
Triggering event	User needs information for ticket purchase or for event attending planning.
Preconditions	Internet access, standard browser, ticket booking application started on web server, ticket database is accessible.
Assumptions	Internet browser software, internet access
Successful end states	List of events, tickets. Message "No suitable event or ticket found".
Unsuccessful end states	No internet access. Ticket database is not available. Too many users work with database (database is busy).
Normal Course	User chooses Ticket Search option FR1: Application must have this option (button or link).
	User chooses Ticket Searching criteria (ticket database connection (it should be in parameters data), date, event name, maximal ticket price, event place, event type, max.returned results number, etc.) FR2: Application must have possibility to fill search criteria.
	System returns list of events or list of tickets for events. FR3: Application must represent returned data in list or grid format. When user click on rows in list, details are displayed. FR4: For registered and logged user must be possibility to book or purchase tickets directly from list.
Variations	If [no connection to internet] then [application cannot run (browser accessibility message)] (related to Action 1). If [Registered and Logged User] then [User can continue to book or purchase tickets (those options are visible)] (related to Action 3).
Alternatives	If [no ticket database is available] then [application returns message about data availability] If [exotic browser] then [application returns standard info message window and recommends to change browser] If [no internet access] then [application returns standard info message window] If [no ticket data fits to criteria] then [application returns message about and recommends to change search criteria]

Fig. 6. One example KD Software use case specification – specifying the use case *Ticket searching*

The main difference between the KD and CA evaluations was that KD Software had earlier published one service specification in the registries, called *KDTicket-DataDelivery2a*, which could be invoked in the ticketing application. The short description of the service was:

Service provides ticket data delivery in several formats (list, one concrete item, etc.). Service also has operation for data searching.

The KD analyst generated 2 service queries from 2 of the 10 use case specifications entered into UCARE. Both were composed of text extracted from the use case name,

Table 4. Top 10 service specifications retrieved for service queries generated from 2 KD Software application use case specifications

Rank	Ticket Browsing Use Case	Ticket Searching Use Case
1	KDTicketDataDelivery2a	ViaMichelinFindNearByPOIwebservice
2	AGENDAMSD	KDTicketDataDelivery2a
3	ViaMichelinFindNearByPOIwebservice	Weblogwebservice
4	ImageCutOut	WebServiceSearch
5	Weblogwebservice	XgniteEdgar
6	XgniteEdgar	Mobile7NavigationKit
7	EmailVerifier	FoldCalc
8	AmazonHistoricalPricingService	ImageCutOut
9	Anagram	KDfindCarService1
10	CalendarServiceEMIC	ABAEpress

actors, précis, problem statement, assumptions, preconditions and triggering event, and 5 or 6 requirements of different types. Both were specified to search using the noun, verb, adverb and adjective parts-of-speech, but no query expansion was requested because the KD analyst explained in debriefing sessions that he did not understand the meanings of the terms *synonyms*, *hyponyms* and *glosses* in the service query.

The top 10 service specifications retrieved for the 2 service queries are reported in Table 4, ranked by MatchValues computed by EDDiE.

The target service specification - *KDTicketDataDelivery2a* - was ranked in the top two by EDDiE for both service queries. In the first query EDDiE retrieved it with full MatchValue (100). For the second use case that specified ticket searching, EDDiE also retrieved the *ViaMichelinFindNearByPOIwebservice* service with full match value that, according to the KD analyst, had no relation with the generated query. The description of this service, also taken from the SeCSE service registries, was:

the "FindNearbyPOI" Web Service allows searching for a certain number of addresses or locations closest 'as the crow flies' to a particular address or place of interest within a user-definable search radius. Then displaying any detailed poi information is possible. For example, it can look for car dealers closest to a given location or find competitors that are closest to your sales points and, as a result, analyze catchment areas that are the least well served.

The analyst's decision not to select query expansion types in the service query prevented EDDiE from generating additional query terms with the same or similar meaning to original query terms. Therefore EDDiE matched only identical or very similar terms such as *search* and *display*. We conjecture that the match values of the retrieved services including *ViaMichelinFindNearByPOIwebservice* would have been different if query expansion was enabled. This result demonstrated that, for ambiguous and incomplete queries, EDDiE generated false positives during service discovery alongside true positive discovered services.

During step 6 the KD analyst attempted to edit the use case and requirements specifications using information in retrieved services. However the changes made by the KD analyst were simple and did not lead to significant changes to new service queries or retrieved services. At this point the KD analyst ended the evaluation.

7 Research Questions Revisited

We used data from the 2 evaluations to answer the 2 research questions. The answer to Q1 – can SeCSE tools retrieve specifications of services that can implement requirements specified by analysts in service queries – was yes. In the KD Software evaluation EDDiE retrieved the target service specification with rank 1 and 2 of 154 with 2 service queries. This suggests high precision of the EDDiE algorithm in the presence of ambiguous and incomplete requirements in a real-world project. Failure to use query expansion increased the relative weighting of syntactic similarities during service discovery and, in the second query, returned one high-ranked but irrelevant service specifications. In the CA evaluation, for which there were no target service specifications, the analyst retained 6 of the 14 retrieved services to invoke in the application. This suggests that the process and environment has the potential to support applications when sufficient numbers of application-independent services are published. That said, our decision to evaluate the utility of the tools, rather than determine their precision and recall, means that we do not know whether the algorithm failed to retrieve other service specifications that might also have been retained by the analyst.

We investigated post-retrieval changes to the use case and requirement specifications to answer Q2 – can analysts using the SeCSE tools make requirement specifications more complete. There was little evidence to answer yes. The CA evaluation added 2 requirements to and changed 1 of the original 6 requirements, and there were no changes in the KD evaluation. Post-evaluation questions revealed that both analysts encountered difficulties browsing retrieved services due to complexities in the service descriptions and similarities with the requirements in the service queries. Poor expression of non-functional requirements meant that filtering services on quality-of-service compliance could not be used, and the Service Browser provided little support to each analyst to understand services. Furthermore UCaRE did not provide support for service querying that was sensitive to the recent changes to requirement and use case specifications. Because service-based changes were small in the context of the use cases specifications, the revised queries did not retrieve new services.

Of course there are numerous threats to the validity of the reported results, and important ones are reported here. The obvious threat to the validity of conclusions drawn was the small number of studies and both were participants in the SeCSE project. To minimize this threat, follow-on studies with more analysts from organizations external to the project are now taking place, and we will interpret results reported in this paper in light of the results from these future evaluations. One threat to the internal validity of the evaluations was that the application requirements and registry services were (unintentionally) aligned too well – we might not expect such alignment in public, market-oriented registries. However, results from the CA evaluation do not support this threat. The domain-independent nature of the services – for verifying addresses, calculating journey times and computing currency exchanges – made them candidates for invocation, and the SeCSE software tools retrieved and presented these services effectively enough to enable the analyst to retain them. A threat to the evaluation's construct validity also merits a mention here. Because the analysts were SeCSE partners with a vested interest in the outcome we cannot discount that they were biased to generate positive results. And one threat to the external validity of the results was our decision to align SeCSE's requirements process and service discovery tools with

UML. Whilst we chose UML for its ubiquity in software development, it does mean that the evaluation results might be less applicable to projects that adopt workflow and business process approaches to requirements analysis.

8 Discussion and Future Work

Answers to the 2 research questions investigated in this paper indicate future directions of research and evaluation in SeCSE. One is the need for precision-and-recall experiments of the EDDiE algorithm and FrEDDiE, a new software module in the environment that decomposes service queries to increase the likelihood of successful retrieval with coarse-grain use cases and requirements. Controlled variables in these experiments will be predefined service query attributes such as use case précis and requirement descriptions and expansion types such as synonyms and glosses. Application experts will review retrieved services for their relevance to each query to generate precision and recall measures for different query attributes and expansion strategies. We are also extending EDDiE to retrieve other types of services, such as peer-to-peer (P2P) and grid services, thus leveraging new repositories of software services of different types on the Internet. In this extension EDDiE service queries are translated into the Universal Service Query Language [12] then fired at federations of P2P and grid service registries compliant with different standards applicable to these service types. Of course, retrieval of more candidate services from more sources amplifies 2 problems reported in the evaluations, which was how to understand and select between retrieved services, then revise requirements and service queries using relevance feedback from retained services.

The answers to the 2 research questions also provide empirical foundations for further development of the SeCSE service discovery environment, particularly in light of our answer to research question Q2. One priority is to improve the usability of the Service Browser module. To make it more usable we responded to post-review comments from the 2 analysts and developed an off-line version of the module in Microsoft Excel. Analysts can now download all data about service queries, specifications of retrieved services, match values and mappings between terms to interactive spreadsheets, to review the data off-line and manipulate it in other forms more supportive of comprehension and selection tasks.

The low number of requirements generated by both analysts when reviewing the retrieved web services contrasts with the higher number of requirements generated from services during facilitated workshops [4]. This difference indicates the need to improve tool support for analysts during this step. To this end we are developing one new software module and adding new features to a second to support service understanding and selection. The Service Browser module does not provide analysts with explicit support for generating or editing requirements in the UCARE module. Instead the analyst is expected to flip between the 2 modules in a single web browser window, using problem analysis and requirements writing skills to document new or changed requirements in UCARE. Therefore we designed a new software module to generate candidate new requirements descriptions from service specification text highlighted as relevant by the analyst. This new module will use mappings between terms computed by EDDiE in use case and requirement specification and retrieved service specifications to generate candidate descriptions of new requirements structured using requirements writing guidelines [13]. The analyst then selects between

and edits the candidate requirements in UCaRE and links it to the service specification for traceability purposes. Of course, if successful, this requirements auto-generation module could be applied to other sources of requirement-related data such as software product descriptions.

Finally we are also adding a new feature to UcaRE to support the iterative and incremental SeCSE requirements process. In both evaluations the 2 analysts were unable to retrieve further service specifications because requirements changes from relevance feedback were small in the context of the original requirement specification. The new feature will allow an analyst to generate service queries that only include requirement and use case information that is new since the last service query(ies) were fired. We predict that the feature will enable focused searching and service retrieval, a prediction that we will investigate empirically in future user studies with the SeCSE service discovery environment.

Acknowledgements

SeCSE is funded by the EU 511680 Integrated Project.

References

1. Tetlow, P., Pan, J., Oberle, D., Wallace, E., Uschold, M., Kendall, E.: *Ontology Driven Architectures and Potential Uses of the Semantic Web in Software Engineering*. W3C (2005)
2. Margaria, T.: *Service in the Eye of the Beholder*. *IEEE Computer* 40(11), 33–37 (2007)
3. Zachos, K., Maiden, N.A.M., Zhu, X., Jones, S.: *Discovering Web Services To Specify More Complete System Requirements*. In: Krogstie, J., Opdahl, A., Sindre, G. (eds.) *CAiSE 2007 and WES 2007*. LNCS, vol. 4495, pp. 142–157. Springer, Heidelberg (2007)
4. Zachos, K., Maiden, N.A.M.: *Discovering Services to Support Creative Thinking during Early Requirements Processes*. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) *ICSOC 2007*. LNCS, vol. 4749. Springer, Heidelberg (2007)
5. Jones, S.V., Maiden, N.A.M., Zachos, K., Zhu, X.: *How Service-Centric Systems Change the Requirements Process*. In: Pastor, Ó., Falcão e Cunha, J. (eds.) *CAiSE 2005*. LNCS, vol. 3520, pp. 13–14. Springer, Heidelberg (2005)
6. Robertson, S., Robertson, J.: *Mastering the Requirements Process*. Addison-Wesley-Longman (1999)
7. Sawyer, P., Hutchinson, J., Walkerdine, J., Sommerville, I.: *Faceted Service Specification*. In: *Proceedings SOCCER (Service-Oriented Computing: Consequences for Engineering Requirements) Workshop*, at RE 2005 Conference, Paris (2005)
8. <http://vega.soi.city.ac.uk/~cc559/REFSQ2008Figure2.jpg>
9. Miller, K.: *Introduction to WordNet: an On-line Lexical Database Distributed with WordNet software* (1993)
10. <http://vega.soi.city.ac.uk/~cc559/REFSQ2008Figure4.tiff>
11. Deliverable A2.D10 - Evaluation of service discovery environments, v2.0, SeCSE Technical Report, available at secse.eng.it (2007)
12. SODIUM, *Service-Oriented Development In a Unified framework*, IST-FP6-004559 (2007), <http://www.atc.gr/sodium>
13. Alexander, I.F., Stevens, R.: *Writing Better Requirements*. Addison-Wesley, Reading (2002)

Mobile Discovery of Requirements for Context-Aware Systems

Norbert Seyff¹, Florian Graf¹, Paul Grünbacher¹, and Neil Maiden²

¹ Johannes Kepler University, Systems Engineering and Automation, 4040 Linz, Austria
`{ns,fg,pg}@sea.uni-linz.ac.at`

² City University London, Centre for HCI Design, London EC1V 0HB, UK
`n.a.m.maiden@city.ac.uk`

Abstract. Understanding the work context of future system users is essential in requirements engineering. It is particularly crucial when developing ubiquitous systems that react on context changes. This paper discusses the need for in-situ requirements elicitation approaches to build mobile and context-aware systems. We identify three different levels of support: The first level covers contextual techniques without tool support. Second level support is based on existing RE approaches and mobile tools. Third level support utilizes context-aware tools receiving context-specific information to guide analysts in the field. These tools enhance requirements gathering for ubiquitous systems. We present a context-aware tool prototype for on-site scenario walkthroughs and discuss how the underlying scenario-based approach needs to be adapted. Our tool-based approach was tested in an initial evaluation study. Finally, the paper presents requirements for RE approaches supporting ubiquitous system development based on lessons learned from using level II and III tools.

Keywords: Requirements elicitation, scenarios, contextual inquiry, mobile and context-aware systems.

1 Introduction

Technology trends such as ubiquitous computing mean new challenges for software engineering and requirements engineering [8]. Ubiquitous systems provide support for everyday activities and their mobile and context-aware nature is a main challenge that needs to be addressed [23]. In particular, understanding the context of such systems is essential for specifying its requirements. Analysts must consider that end user requirements vary according to context changes and that the future system must adapt accordingly. Fahrmeier *et al.* [6] report that many ubiquitous systems pass laboratory tests, but fail to meet real world user expectations when released into the wild. One main reason is the difficulty for future system end users to describe their specific needs in a certain situation or context. Blomberg *et al.* [3] have pointed out that "...people have only limited ability to describe what they do and how they do it without immediate access to the social and material aspects of their lives."

We use a fictitious museum guide of the future as the example throughout this paper to illustrate the unique characteristics of mobile and context-aware systems. In

this example, the museum visitors' aim is still the exploration of exhibitions and artefacts. However, ubiquitous computing technologies in the museum significantly change this everyday activity and allow visitors to retrieve context-specific information via mobile devices and panels on the wall. Museum visitors receive information about ongoing exhibitions, events, and detailed floor plans based on their context, such as their position in the museum. For example, a visitor entering a certain exhibition expects the museum guide to display detailed information on the exhibition's theme. In contrast, visitors entering the museum's cafeteria might want to take a look at the menu. The context-aware museum guide supports the activities of visitors by providing just the right information at the right time and location. The example shows that the needs of visitors vary depending on their current context which changes continuously due to visitors' mobility.

The elicitation and specification of requirements for such systems differs considerably from conventional systems. Neglecting ubiquitous systems' technologies in requirements engineering (RE) is risky as the key characteristics of the future system can easily be overlooked. We propose that RE methods and tools should benefit from the manifold and omnipresent mobile and context-aware technologies. As a result (i) context-aware RE approaches will provide better support for developing ubiquitous systems; and (ii) existing RE approaches will benefit from ubiquitous technologies. For example, context-aware RE tools will significantly enhance the range of possibilities for requirements engineers. Such tools can guide and support them by providing essential information just at the right time in a context-aware manner.

In our research we want to address the problems and challenges of requirements elicitation for ubiquitous systems. In particular, we focus on requirements gathering for mobile and context-aware systems. This work is based on scenarios, a widely and successfully used technique to discover requirements for software systems [12]. In previous studies [13, 15, 16] we used scenario-based techniques for gathering requirements in the work-context of future system users. Based on this work we designed an approach, which uses ubiquitous technologies to support requirements discovery for mobile and context-aware systems. In particular, we developed a scenario walkthrough tool prototype that automatically activates scenario events like "The visitor enters the museum" or "The visitor stands in front of an artefact" based on position signals or tags installed in the museum's environment. We expect that the automatic highlighting of scenario events will increase guidance and thereby lower the hurdles for using such tools.

The work presented in this paper is based on our research on mobile tools for scenario-based RE [13, 14, 15, 16, 20]. We propose a framework that covers three different levels of support for contextual requirements elicitation. According to the framework we present a prototype of a context-aware tool which improves guidance for on-site analysts. We explain how a scenario-based approach can be tailored to support contextual requirements elicitation. Based on a literature review and lessons learned from using our tools we identify requirements for elicitation approaches aiming to support mobile and context-aware system development.

The paper is structured as follows: Section 2 presents a 3-level framework of support for contextual requirements elicitation, introduces basic notions of context and context-awareness, and discusses related work. Section 3 discusses the ART-SCENE method and shows how it can be applied to support contextual requirements elicitation.

Section 4 describes a mobile user-driven requirements elicitation tool based on ART-SCENE. Section 5 describes our context-aware tool prototype. Section 6 discusses requirements for elicitation approaches and tools supporting mobile and context-aware system development. We end the paper with conclusions and future work.

2 Levels of Support for Contextual Requirements Elicitation

The terms context and context-awareness are often used intuitively. However, finding useful definitions is not easy and has led to many debates in the research community [5, 18, 19]. We adopt the definition of Dey *et al.* [5] who define context as "... information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." These entities are highly relevant in requirements discovery for ubiquitous and interactive systems that rely heavily on such context information. The term context-awareness is used to describe software systems, which are able to adapt themselves to their context. Dey *et al.* [5] call a system context-aware if "... it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."

Reasoning about context in RE and system design is not new. For example, Potts *et al.* [17] highlight the influence of location on requirements and on the acceptance of a system in their work on the Inquiry Cycle. Sutcliffe *et al.* [22] describe a framework for contextual requirements engineering, which highlights the dependency between requirements, location, time, and the mutability of requirements by different users. Related research also covers requirements monitoring where changing runtime behaviour triggers system adaptation and reconfiguration [7].

The research presented in this paper is strongly influenced by the ideas and concepts of contextual inquiry (CI) [2]. CI is the first part of contextual design – a customer-centred process that supports understanding users in their workplace and examining how people work to meet real world requirements. CI supports analysts in observing peoples' work and asking questions feeding the design process.

When using CI to elicit requirements for the context-aware museum guide an analyst directly observes visitors in the museum and asks questions about their typical behaviour to better understand their needs as a result of being in the context. CI is based on the four principles context, partnership, focus, and interpretation. Understanding the context of future system users is considered as essential for optimal re-design of work. Partnership means that future system users should be treated as real partners in designing the new system by helping them to articulate their work experience. Focus defines the analyst's point of view, which should steer conversations to reveal details of work. Finally, the right interpretation of gathered data and information is essential for designing a new system satisfying future system user requirements. Although CI and similar approaches provide a good start they generally lack tool support and are not well integrated with existing RE approaches. Researchers highlight this lack of tool support as a major issue in mobile and context-aware system development [8].

Our research on contextual requirements elicitation is based on understanding the context and interacting with the context using appropriate tools in the field. Going

beyond CI our aim is to bring mature and established RE approaches into the workplace of future system end users and to provide tool support for analysts applying them. On-site analysts can understand stakeholder needs more easily by having immediate access to their work environment, including the social and material aspects of their lives. Gathering implicit and tacit knowledge by observing stakeholders and asking questions enables analysts to discover requirements that reflect stakeholders' needs. This includes in particular the discovery of requirements that vary on context changes. Therefore analysts need to be equipped with mobile RE tools that unobtrusively support their on-site analysis. This includes tools that interpret context information and provide intuitive guidance and support for the analysts.

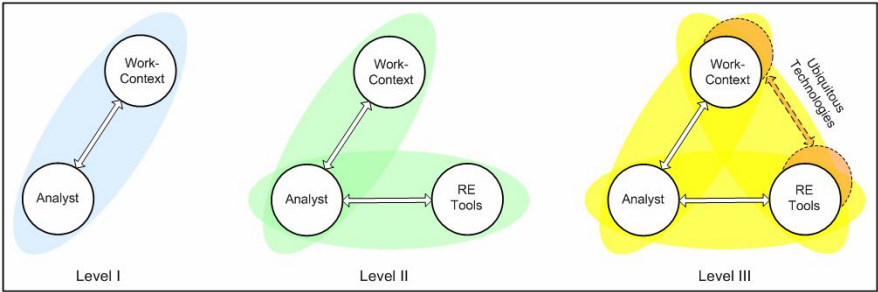


Fig. 1. Levels of support for and elements of contextual requirements elicitation

We identified three key elements of contextual requirements elicitation and dependencies among them (see Figure 1): the work context, the analyst, and the RE tool. The dependencies between these elements highlight the interaction and information flow. In the left part of the figure the analyst observes and interacts with the work context in the domain under analysis. The middle area of the figure visualizes the analyst additionally equipped with a mobile RE tool to support his activities on-site. In the right part of the figure the mobile RE tool uses context-aware technologies to interact with and reveal further information about the work context. We identified three different levels of support for contextual requirements elicitation based on these three key elements and their dependencies (discussed in more detail in Table 1):

Level I represents conventional approaches for understanding context, such as CI. Analysts using these approaches typically move around freely in the work context of future system users. No software tool support is required.

Level II represents user-driven RE tools enabling the requirements engineers to elicit requirements in the work context when moving around freely. Such tools are based on mature and proven RE approaches to support the work of engineers in situ. The tool user is in charge of identifying the current work context. Examples are the Mobile Scenario Presenter (MSP) [16] or ARENA-M [20]. These tools are available on mobile devices, such as Personal Digital Assistants (PDAs) or Smartphones.

Level III context-aware RE tools are capable of identifying the current work context. Based on information received from the context they improve guidance and support for requirements engineers. For example, these tools offer features to inform its users about context changes. An example is the context-aware MSP prototype, which we will present in Section 5.

Table 1. Levels of support for contextual requirements elicitation

	<i>Level I: No tool support</i>	<i>Level II: User-driven tools</i>	<i>Level III: Context-aware tools</i>
<i>Characteristics</i>	Structured dialogue in situ / Guidance based on principles and checklists	Usage of RE approaches in situ / Guidance provided by tool	Usage of RE approaches in situ / Advanced guidance and contextual information provided by tool
<i>Prerequisites</i>	Knowledge of and experience with the method / No technology requirements	Knowledge of and experience with the RE approach and tool / Tool specific technology requirements	Knowledge of and experience with the RE approach and tool / Tool and work context specific technology requirements
<i>Examples</i>	Contextual inquiry	Mobile Scenario Presenter and ARENA-M	Context-aware MSP

Table 1 also shows prerequisites for the different levels of tool support. All three levels rely on knowledge of and experience with the underlying approach. Level II and III further have technology-dependent requirements. User-driven RE tools available on PDAs or similar mobile devices may require wireless connectivity. Context-aware RE tools further depend on technologies enabling them to acquire context information. This includes ubiquitous technologies (e.g., RFID) and context frameworks and middleware [11].

The three levels of tool support for contextual requirements elicitation build on each other. Following the principles of CI user-driven and context-aware tools enable analysts to use mature RE approaches in the work context [16]. The presented levels of tool support are not limited to particular RE methods such as scenario-based approaches. Many existing RE approaches can be complemented with contextual aspects to guide RE work. In the following we show how user-driven and context-aware tools can bring a scenario-based approach in the work context of future system users and discuss challenges of adapting the approach.

3 Tailoring ART-SCENE to Contextual Requirements Elicitation

We developed and evaluated level II and level III tool support for ART-SCENE¹, a scenario-driven approach to discover and analyze requirements [12]. ART-SCENE supports analysts and stakeholders to walk through scenarios in workshops. Scenarios are automatically generated from use case specifications and offer recognition cues by providing alternative courses for each normal course event. These so called what-if questions originate from a model of abnormal behaviour, which covers different types of failures. The idea underpinning these walkthroughs is simple: people are better at identifying errors of commission rather than omission [12]. For example, if an alternative course what-if

¹ Analysing Requirements Trade-offs: SCENario Evaluations.

question is relevant to the system under development stakeholders try to find new requirements addressing the issue behind the alternative course.

The ART-SCENE environment includes several tools. The web-based Scenario Presenter supports a team of analysts and stakeholders to walk through the generated scenarios collaboratively in a face-to-face meeting. In our museum guide example several potential museum visitors are invited to an ART-SCENE workshop. Guided by an analyst they use scenario events, such as “The visitor enters the museum” to discover new requirements for the system. The alternative course what-if questions (e.g., “What-if the mobile device is not working properly in this situation?”) trigger stakeholders to explore abnormal and unusual system behaviour to identify requirements not yet handled in the specification. A scribe using the Scenario Presenter documents the discovered requirements. To attend these workshops stakeholders must leave their work environment. The missing contextual information might limit their ability to report hidden and tacit knowledge, which potentially reduces the effectiveness of the elicitation technique [13].

To overcome existing limitations we developed contextual tool support for ART-SCENE. Our aim is not to replace existing workshops but to complement them with on-site scenario walkthroughs. Before presenting level II and III tool support for ART-SCENE we discuss how the approach needs to be changed to support on-site walkthroughs. Our initial experiences with contextual requirements elicitation tools suggest several changes to ART-SCENE. In particular, we identified four new on-site activities that extend the existing approach (see Figure 2). In situ *scenario validation* and *walkthrough* are the two key activities, which are relevant for all three levels of support and can even be done using paper-based scenarios. We do however not favour this approach for usability reasons. *Prepare environment* and *link context information to scenario elements* are activities needed for level III.

We recommend proceeding as follows:

Generate scenario. The first activity of the analyst is using ART-SCENE to automatically generate scenarios based on use case models. These generated scenarios include normal course events and alternative course what-if questions. This first step is also part of other ART-SCENE projects not considering contextual activities. The generated scenarios are highly relevant for contextual requirements elicitation as they provide a model of the context usable to relate scenario events to the real-world context events.

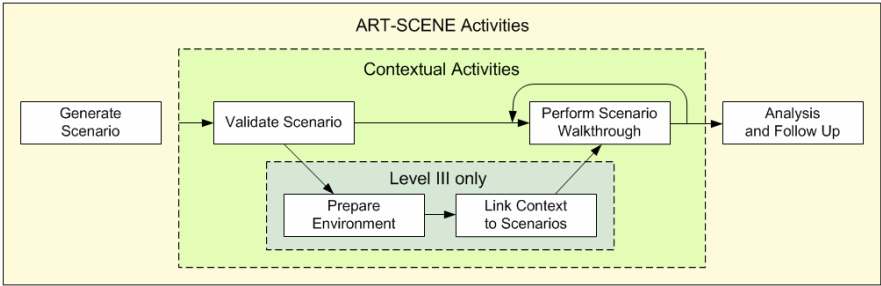


Fig. 2. ART-SCENE and contextual activities

Validate scenarios in context. The ART-SCENE approach supports developing use case models and scenarios [12]. We recommend that the on-site analyst validates the scenarios in the work context to check whether the defined normal course events reflect the observed activities. For example, a generated normal course event sequence for the museum example may contain an event for buying a ticket. If the analyst observes, however, that the museum offers free entrance he may decide to drop the generated event to streamline the subsequent walkthrough. The analyst can also validate alternative course what-ifs to prune irrelevant questions or to add new questions. Level II and III ART-SCENE tools can support the on-site scenario validation.

Prepare environment. Context-aware elicitation tools rely on external information to determine the context of the tool user. However, in many cases the existing environment does not provide sufficient context triggers and it is necessary to prepare it by installing tags or sensors. In our museum example the analyst might decide to install tags in selected areas of the museum, which later allow determining the user's context, i.e., his position in the museum. Depending on the actual situation in the work environment and the problem at hand the analyst carefully chooses the appropriate ubiquitous technology (this step is only necessary when using a level III RE tool).

Link context information to scenario elements. After preparing the environment the external triggers need to be linked to scenario elements. This enables a context-aware level III tool to highlight relevant scenario events automatically. A trained analyst with sufficient domain knowledge and tool experience links domain and scenario events and thereby shares his domain knowledge with other analysts or even future system end users. In our museum example a senior analyst identifies and analyses possible external triggers for the scenario event "The visitor enters the museum" when stepping inside the museum. He then links the external triggers to the scenario event. Another analyst benefits from this information as the context-aware tool can guide him by highlighting the event automatically as soon as he enters the museum. In many cases there will be multiple external triggers for one event. This is especially relevant for alternative course what-if questions. For example, the alternative course event "What if the museum is closed?" could be triggered initially by the same signal provided for the normal course event "The visitor enters the museum" but would also need a second trigger based on time constraints, such as the museum's opening hours. It is however important to note that not all normal course and alternative events need to be covered by level III support. As the three levels of contextual requirements elicitation build on each other some events can also be addressed by level I and level II support.

Perform scenario walkthrough. The analyst performs a contextual scenario walkthrough in situ following the principles of contextual inquiry. He observes current system behaviour and asks questions to future system users to discover requirements. The analyst is supported by a user-driven level II or context-aware level III tool. A level III tool provides additional guidance and information and automatically recommends actions to the analyst as soon as the context changes. This feature allows the analyst to unobtrusively interact with stakeholders as the tool determines the current context.

New requirements for the context-aware museum guide will be discovered and captured by analysts observing context changes when walking around in a museum. For example, the requirement "The museum guide shall provide information about

exhibitions” is gathered for the normal course event “The visitor enters the museum”. Another requirement “Inform visitors about the estimated waiting time” is identified when the on-site analyst waits in line to enter the museum. Depending on the level of support the analyst either selects the appropriate events himself (level II) or the tool will support this task (level III).

Analysis and Follow Up. The main aim of the on-site analyst is to interact with future system users in an unobtrusive and uninterrupted way. High mobility and frequent context changes in the field limit the time for documenting requirements. This strengthens the need for level III tool support and requires easier ways to document requirements. Instead of full requirements analysts are advised to record information cues instead [13]. To support this way of requirements documentation level II and level III need to provide features for the fast recording of recognition cues, such as audio. After the walkthrough the analyst can use ART-SCENE’s desktop tools to analyze the gathered information and to specify detailed requirements.

Earlier experiences show that by following this tailored ART-SCENE approach analysts are able to gather requirements in the field with potential benefits for requirements correctness and completeness [13, 15]. We assume that several analysts and stakeholders will be involved in on-site requirements discovery. This means that on-site scenario walkthroughs will be repeated several times. This assumption is especially relevant for level III elicitations, as the preparation steps, such as setting up the environment and linking external triggers to scenario events are time consuming and challenging. The presented activities extend the existing ART-SCENE approach to support on-site contextual requirements elicitations. In the next sections we introduce tools using this approach to support on-site analysts.

4 The Mobile Scenario Presenter: A Level II Tool

The Mobile Scenario Presenter (MSP) is a PDA-based mobile scenario walkthrough tool providing level II support for ART-SCENE [15, 16, 20] by making selected capabilities of ART-SCENE available to mobile analysts discovering requirements in the workplace of future system users. The MSP is based on contextual inquiry and supports its key principles [14]. The on-site analyst uses the MSP when observing current work context and interacting with future system users. Working in the field the analyst is able to gather context specific requirements. The MSP is a user-driven level II tool and does not interact with the context. The generated alternative course what-if questions guide the analyst when asking questions about abnormal system behaviour in different contextual situations. The discovered requirements can be documented using PDA specific multimedia capabilities. In our museum example an analyst equipped with the MSP discovers and captures requirements while walking around in the museum. The analyst observes people’s behaviour and asks questions according to the scenarios what-ifs. Further the MSP’s audio recoding feature is used to document requirements.

We performed several case studies to empirically test the usefulness and usability of the tool [15]. At Belfast City Airport the MSP was used to gather requirements for the air traffic management system VANTAGE. A detailed discussion of this evaluation can be found in [13]. This evaluation revealed that analysts using the MSP

discovered different requirements than compared to workshop scenario walkthroughs. In particular, the analysts documented numerous location-specific requirements regarding departure gates, air bridges, and dispatch offices showing the importance of context for requirements discovery. For instance, although detailed domain modelling had been done prior to the mobile walkthrough the analysts using the MSP discovered two new actors. Another interesting outcome was that the on-site analysts were able to document significantly more requirements per hour of stakeholder involvement as compared to workshop scenario walkthroughs.

Previous evolutions discovered that it is difficult for a single analyst to work with the tool and interact with stakeholders simultaneously in highly dynamic environments [15]. Therefore in VANTAGE the more experienced analyst took the role of the facilitator and the other analyst operated the MSP as a scribe. The idea was that facilitator and scribe observe on-site activities. The facilitator's task was to ask questions to airport and airplane staff. The scribe documented requirements and communicated the information provided by the MSP to the facilitator. However, the majority of the gathered requirements were triggered by the work context and not by scenario events provided by the MSP. Possible reasons are the richness of work context triggers, the experience of the facilitator, the fact that the facilitator could hardly see the provided what-if questions, as well as the limited communication between the facilitator and the scribe during ongoing interviews.

In a second project the MSP was used to gather requirements for the APOSDLE system [9]. The APOSDLE system provides individual learning support for information workers and features to contribute new content to an organisation's knowledge pool. Two analysts visited stakeholders in offices in Graz and Dortmund to gather requirements for this system. This time both analysts were equipped with a PDA running the MSP based on the VANTAGE lessons. This enabled the analysts to follow the scenarios provided by the MSP and to switch the role of facilitator and scribe. Both analysts accessed the same database to synchronize their activities. As in the VANTAGE project the analysts intensively used the audio recording feature of the MSP. One interesting observation during this project was that the two analysts sometimes interpreted the real world context differently. For example, both analysts gathered the same requirement but added it to a different scenario event. Another outcome was that the start and end events provided by the MSP (see Figure 3) helped the analysts to handle context changes. Rotating the role of facilitator and scribe made both analysts aware of the what-if questions. Although the time for handling the tool was short it still affected the analysts' ability to interact with stakeholders. Using the MSP the analysts had to be familiar with the tool, the domain, and the scenario events to work effectively. To provide additional guidance for analysts we started to evolve the MSP into a context-aware application.

5 The Context-Aware MSP: A Level III Tool

Based on the MSP we developed a context-aware level III tool prototype by enhancing the level II MSP with a capability for receiving infrared signals from other devices to detect context changes. By interpreting the received contextual information the prototype improves guidance of analysts by automatically highlighting the actual scenario event.

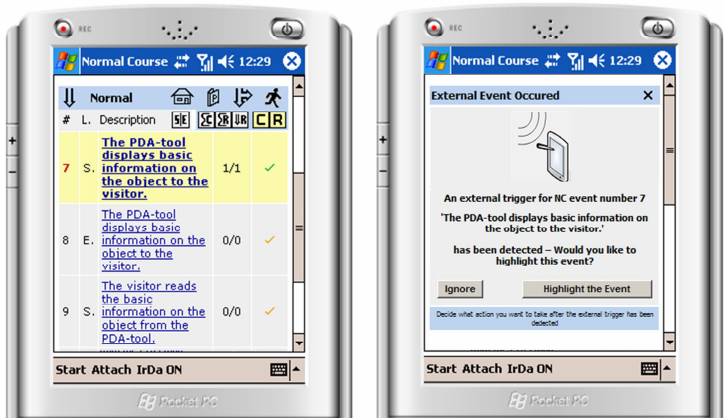


Fig. 3. Level III tool normal course (left) and event highlighting (right)

The level III MSP prototype has been used in an initial evaluation study to explore the impacts of context-awareness on scenario-based RE. We positioned several laptop computers with activated infrared in our laboratory together with artefacts and pictures to simulate relevant parts of a museum (see Figure 4). The unique identifiers transmitted via infrared referred to normal and alternative course events of the scenario. Linking the external triggers to scenario elements was done manually in the underlying ART-SCENE database. This enabled the MSP to determine the users' position in the fictitious museum. As soon as the tool received a new infrared signal it informed the user that his context had changed and highlighted the related normal or alternative course event (see Figure 3). For example, if a user approached the entrance, the scenario event "The visitor enters the museum" was automatically highlighted. The user then captured requirements for this particular context using the audio recording feature.

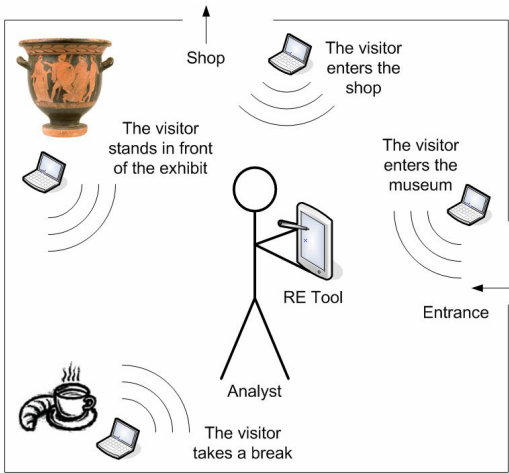


Fig. 4. Using the context-aware scenario tool in the museum

We performed several scenario walkthroughs in the laboratory museum to gain first experiences with the level III tool. Some of the participants already had used the level II tool in real word settings and had experienced the need of continuously interacting with stakeholders. A second group of analysts had limited experience in using the MSP. Despite the restrictions of infrared (e.g., the PDA must be in line of sight to the laptop to trigger context changes) the level III tool was able to provide relevant context-information. All analysts were successfully guided through the scenario and none had to use event selection and scrolling functions, a feature that had led to some problems in earlier evaluation studies with the MSP [15]. The analysts documented requirements to the automatically highlighted events using the audio recording feature. Some analysts with limited MSP experience reported that handling the audio recording feature reduced their ability to focus on the work context. The more experienced analysts did not report this issue. One reason could be that they were already used to the MSP's user interface. Another analyst requested an audio alarm feature to inform him about context changes.

6 Requirements for RE Approaches

We present initial requirements for RE approaches and tools addressing the characteristics of mobile and context-aware systems based on lessons learned from the in-situ use of the level II MSP, the initial feasibility study of the context-aware MSP, and a literature review. As mobility and context-awareness are essential for ubiquitous systems [1, 10, 23] the identified requirements are also relevant for ubiquitous system development. In the following paragraph we discuss the identified requirements by highlighting their rationale:

Usable on-site. To discover requirements for a mobile and context-aware system it is essential to understand the existing work environment. Working on-site enables analysts to observe and interact with the current work context with potential benefits for requirements completeness and correctness [2, 3, 15].

Support mobility. This requirement covers two aspects of working on-site: First, if mobility is an essential part of the system under development [1] it must be addressed during requirements discovery [4]. The second aspect covers the mobility of the analyst in the work context. In many cases the analyst needs to move around freely without any restrictions to observe stakeholder activities on-site. For example, when discovering requirements for VANTAGE the analyst had to enter the cockpit of an airplane to interview the pilot [13].

Usable unobtrusively. The analyst in the context might influence the work of stakeholders to be observed. To avoid this influence the analyst needs to use approaches and tools supporting him without distracting future system users. While PDA-based tools supported these requirements in most cases we experienced that elderly people at London bus stops were distracted by this kind of device [15].

Provide a model of the work context. Researchers highlight that any context-aware system relies on a well-defined context model [21]. Such a model is also needed when

eliciting requirements for such systems as it increases focus and enables analysts to link discovered requirements to modelled contextual situations.

Consider unusual or unexpected system behaviour. Approaches such as ART-SCENE provide recognition cues in the form of what-if questions to discover requirements for unusual or unexpected system behaviour [12]. Working in the context might trigger even more requirements than the what-if questions [13]. Nevertheless, as the name suggests, unusual contextual situations are rarely experienced by the on-site analyst. The what-if questions are thus highly relevant for on-site analysts.

Detect context changes automatically. During interviews the analyst’s attention is on interacting with stakeholders [2]. This limits his ability to determine the actual context in dynamic environments. Detecting context change automatically improves guidance and lowers the hurdles for analysts to work on-site.

Usable by end users. Support for end users requires tools that provide guidance when walking through scenarios. This requirement also stresses the need to minimize user interaction.

Using these requirements Table 2 summarizes the difference between CI, the level II MSP, and the level III context-aware tool with respect to the specific way of support. Being in the work context is a prerequisite for CI and the MSP tools. The approaches and tools support mobility and can be used unobtrusively. Based on ART-SCENE the MSP tools provide a context model in the form of scenarios and recognition cues. The context-aware MSP prototype provides more guidance for on-site analysts. It detects context changes and refers this information to the underlying model using ubiquitous technologies. The currently available tools are still inadequate to be used by end users as the degree of guidance is still insufficient.

Table 2. Characteristics of contextual requirements elicitation approaches

	<i>Level I: Contextual Inquiry</i>	<i>Level II: Mobile Scenario Presenter (MSP)</i>	<i>Level III: Context-aware MSP</i>
<i>Can be used on-site?</i>	✓	✓	✓
<i>Supports mobility?</i>	✓	✓	✓
<i>Unobtrusive?</i>	✓	✓	✓
<i>Provides model of context?</i>	-	✓	✓
<i>Considers unexpected behaviour?</i>	-	✓	✓
<i>Context change detected?</i>	-	-	✓
<i>Usable by end users?</i>	-	-	-

7 Conclusions and Future Work

The increasing use of mobile and context-aware systems will require novel approaches in requirements engineering. In this paper we discuss a framework that identifies three layers of support for contextual requirements engineering. The MSP is an example of a level II user-driven approach. We presented a novel prototype of a

context-aware tool offering level III support by interpreting and reacting to contextual information provided via infrared signals. The impact of these tools on the underlying ART-SCENE approach is discussed as an example how established approaches can be tailored to the needs of on-site work. We summarize our lessons learned by identifying requirements for approaches supporting the discovery of requirements for mobile and context-aware systems.

Our tool prototype shows the feasibility of developing requirements elicitation approaches, which use ubiquitous technologies to identify the current work context.

Further research challenges include, but are not limited to the following:

Extending the notion of context is highly relevant for further evaluation studies. In our study the notion of context was limited to a single contextual aspect (physical location). We plan to extend this notion to explore the effects of receiving several contextual triggers (e.g., time, weather, and social context).

The process of setting up the environment needs to be explored to optimize the benefits and reduce the costs of the approach. As discussed in section 3 it is necessary to link these triggers to scenario elements before working with the context-aware tool. We need to experiment with technologies that can easily be installed in environments to start the elicitation process. In certain cases this step might be obsolete if ubiquitous technologies are already present.

The process of linking scenario events and real world scenes needs more research. In particular, finding appropriate external trigger for all alternative course what-if questions can be challenging. We assume that in many cases it will not be possible to identify an external trigger for each what-if question. Grouping of alternative course events might be a solution.

Exploring the possibility of adaptive scenarios that change their appearance depending on the context is another research issue. We are envisioning scenarios that adapt according to the context, e.g., by changing the ordering of events. These normal course variations would better support on-site analysts. We experienced that in many cases the work processes of future system end users vary slightly depending on several criteria, e.g., the weather. For example, when discovering requirements for the VANTAGE systems [13] the analysts observed that bad weather conditions affected the work of the airport staff. Evolved level III context-aware tools providing adaptive scenarios could be helpful for on-site analysts in such situations.

Supporting future system end users in documenting their needs is another important part of further research. We believe that evolved level III tools will not only support analysts but will also enable future system end users to document their needs themselves. For instance, end users could document their requirements using the audio recording feature automatically guided through scenarios by the MSP. Later these recognition cues could be transcribed into requirements by an analyst.

References

1. Abowd, G.D.: Software engineering issues for ubiquitous computing. In: Proc. 21st IEEE Intl. Conference on Software Engineering. IEEE CS, Los Angeles (1999)
2. Beyer, H., Holtzblatt, K.: Contextual Design: Defining Consumer-Centered Systems. Morgan-Kaufman, San Francisco (1998)

3. Blomberg, J., Burrell, M., Guest, G.: An ethnographic approach to design. In: Jacko, J.A., Sears, A. (eds.) *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*. Lawrence Erlbaum Associates, Mahwah (2002)
4. Cheverst, K., Davies, N., Mitchell, K., Friday, A., Efstratiou, C.: Developing a context-aware electronic tourist guide: some issues and experiences. In: *Proc SIGCHI Conference on Human factors in computing system*. ACM, The Hague (2000)
5. Dey, A.K., Abowd, G.D., Brown, P.J., Davis, N., Smith, M., Steggle, P.: Towards a Better Understanding of Context and Context-Awareness. In: *Proc 1st Intl. Symposium on Handheld and Ubiquitous Computing*. Springer, Karlsruhe (1999)
6. Fahrmaier, M., Sitou, W.: Unwanted Behavior and its Impact on Adaptive Systems in Ubiquitous Computing. In: *14th Workshop on Adaptivity and User Modeling in Interactive Systems*, Hildesheim (2006)
7. Fickas, F., Feather, M.S.: Requirements monitoring in dynamic environments. In: *Proc 2nd IEEE Intl Symposium on Requirements Engineering*. IEEE CS, New York (1995)
8. Fouskas, K., Pateli, A., Spinellis, D., Virola, H.: Applying contextual inquiry for capturing end-users behaviour requirements for mobile exhibition services. In: *1st Intl. Conference on Mobile Business*, Copenhagen (2002)
9. Ghidini, C., Pammer, V., Scheir, P., Serafini, L., Lindstaedt, S.: APOSDLE: learn@work with semantic web technology. In: *I-Know 2007*, Graz (2007)
10. Kindberg, T., Fox, A.: System software for ubiquitous computing. *IEEE Pervasive Computing* 1(1), 70–81. IEEE EAD, Piscataway (2002)
11. Leaver, S.C., Mendelsohn, T., Overby, C.S., Yuen, E.H.: Evaluating RFID Middleware. *RFID Journal* (September 2004)
12. Maiden, N.: Systematic Scenario Walkthroughs with ART-SCENE. In: Alexander, I., Maiden, N. (eds.) *Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle*. John Wiley, Chichester (2004)
13. Maiden, N., Ncube, C., Kamali, S., Seyff, N., Grünbacher, P.: Exploring Scenario Forms and Ways of Use to Discover Requirements on Airports that Minimize Environmental Impact. In: *Proc 15th IEEE Intl. Requirements Engineering Conference*. IEEE CS, New Delhi (2007)
14. Maiden, N., Seyff, N., Grünbacher, P.: The Mobile Scenario Presenter: Integrating Contextual Inquiry and Structured Walkthroughs. In: *Proc 13th IEEE Intl. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. IEEE CS, Modena (2004)
15. Maiden, N., Seyff, N., Grünbacher, P., Otojare, O., Mitteregger, K.: Making Mobile Requirements Engineering Tools Usable and Useful. In: *Proc 14th IEEE Intl. Requirements Engineering Conference*. IEEE CS, Minneapolis (2006)
16. Maiden, N., Seyff, N., Grünbacher, P., Otojare, O., Mitteregger, K.: Determining Stakeholder Needs in the Workplace: How Mobile Technologies Can Help. *IEEE Software* 24(2), 46–52. IEEE CS, Los Alamitos (2007)
17. Potts, C., Takahashi, K., Anton, A.I.: Inquiry-based requirements analysis. *IEEE Software* 11(2), 21–32 (1994)
18. Schilit, B.N., Adams, N., Want, R.: Context-Aware Computing Applications. In: *Proc Workshop on Mobile Computing Systems and Applications*. IEEE CS, Santa Cruz (1994)
19. Schmidt, A., Aidoo, K.A., Takaluoma, A., Tuomela, U., Laerhoven, K.V., Velde, W.V.d.: Advanced Interactions in Context. In: Gellersen, H.-W. (ed.) *HUC 1999*. LNCS, vol. 1707, pp. 10–89. Springer, Heidelberg (1999)
20. Seyff, N., Grünbacher, P., Maiden, N., Tosar, A.: RE Tools Go Mobile. In: *Proc 26th IEEE Intl. Conference on Software Engineering*. IEEE CS, Edinburgh (2004)

21. Strang, T., Linnhoff-Popien, C.: A Context Modeling Survey. In: Workshop on Advanced Context Modelling, Reasoning and Management, Nottingham (2004)
22. Sutcliffe, A.G., Fickas, S., Sohlberg, M.M.: PC-RE: a method for personal and contextual requirements engineering with some experience. *Requirements Engineering* 11(3), 157–173 (2006)
23. Weiser, M.: The Computer for the 21st Century. In: Baecker, R.M., Grudin, J., Buxton, W.A.S., Greenberg, S. (eds.) *Human-computer interaction: toward the year 2000*. Morgan Kaufmann, San Francisco (1995)

When to Adapt? Identification of Problem Domains for Adaptive Systems

Kristopher Welsh and Pete Sawyer

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

Abstract. Dynamically adaptive systems (DASs) change behaviour at run-time to operate in volatile environments. As we learn how best to design and build systems with greater autonomy, we must also consider when to do so. Thus far, DASs have tended to showcase the benefits of adaptation infrastructures with little understanding of what characterizes the problem domains that require run-time adaptation. This position paper posits that context-dependent variation in the acceptable trade-offs between non-functional requirements is a key indicator of problems that require dynamically adaptive solutions.

Keywords: Adaptive systems, non-functional requirements.

1 Introduction

Kephart and Chess [1] identified the move to autonomic computing as a grand challenge to the software engineering community. They argue that systems able to monitor, (re)configure, (re)construct, heal and tune themselves at run-time, are needed to mitigate the ever increasing size and complexity of computing systems; which are expected to operate in ever less predictable and stable environments. Although such systems remain out of reach today, important steps toward them are being taken by the research community with self-managed, or dynamically adaptive systems (DASs). A DAS alters its behaviour or composition in response to changes in its environment.

All software systems have to cope with changes in their environment, but usually the environment changes slowly enough for adaptation to be performed off-line. Web browsers, for example, need to adapt to cope with new content types and protocols with the development of new versions that can be installed as updates on users' computers. Increasingly, however, systems are being conceived that need to adapt at run-time. For example, applications at the "wireless edge" of the Internet, must adapt to the fluctuating availability of services as users move between areas covered by different networks. Other examples include systems adapting to cope with different user needs [2], new network topologies [3][4], and radical change in physical environments [5].

There are two common types of adaptation: parametric and architectural [6]. Parametric adaptation involves building adaptive capabilities into code on a per-application basis, radically increasing complexity and making DASs costly to build and maintain. Architectural adaptation, by contrast, uses an adaptive infrastructure

which typically effects adaptation by component substitution without suspending execution [7][8]. Adaptive complexity is partitioned to a reusable and configurable adaptive infrastructure, easing the maintenance of applications that use it. Dynamic adaptation is a technology that is still maturing and many of the DASs reported in the literature have been developed to showcase the capabilities of particular adaptation infrastructures.

Although adaptive infrastructures provide a mechanism for easing implementation complexity at the implementation level, the complexity inherent in the problems for which DASs provide a solution remains a challenge. As the enabling technology continues to mature, we will need to improve our understanding of how to analyse, specify and design DASs, so that we can cope with the conceptual complexity posed by volatile environments. At the requirements level, Berry et. al. [9] have identified four levels of RE needed for DASs, which has been used as the basis for subsequent work on goal-driven analysis of DASs [10][11], along with other approaches investigating their requirements: e.g. [12]. In most cases, the RE for DASs start with an assumption that the problem under analysis requires a DAS as the solution, and that therefore the need for dynamic adaptation is somehow obvious from the outset. There may well be families of problems where this will be true, but it may not always be clear. Such ambiguity risks over-engineering systems for which dynamic adaptation is not, in fact, a requirement. Similarly, failure to recognize the presence of such a requirement early in a project may result in cost underestimation or worse.

In this position paper, we posit that a problem that requires a DAS will exhibit identifiable characteristics that if not present, strongly indicate that a conventional, static system will provide an adequate solution. If our hypothesis holds true, it should act as a litmus test usable for analysts during the early-phases of RE.

2 Volatile Problem Domains, Adaptive Requirements

For our purposes here, we consider the requirement to adapt dynamically to be imposed by the environment in which the system must operate. In general, we exclude systems that use adaptation as a defensive strategy to cope with design or implementation failures, perhaps by adopting a ‘limp-home’ mode on detection of a failed component. An exception to this rule is where the system is designed to cope with failure conditions that have their root in a ‘failure’ of the analysis process to anticipate possible states of the environment. In our terms, dynamic adaptation is a legitimate mitigating strategy when the analyst recognizes that their model of the environment is incomplete. For example, there may unknowable properties of the atmosphere of Mars that the designers of a probe nevertheless need to try to cope with.

Systems that must cope with unknowable environments are at the extreme end of a spectrum of DASs. More common is the situation where the environment is volatile but understood sufficiently well to allow the analyst to anticipate how it will change. Here, the approach advocated by Berry et al [9] is to characterize the environment as a set of discrete stable *domains* that it can transition between. A DAS can then be conceptualized to comprise a set of *target systems*, each designed to operate within a domain. The analyst’s job is then to specify each target system and the *adaptation*

scenarios [10][11][13] that specify when the system adapts from one target system to another.

The question we seek to answer is how can a requirement for dynamic adaptation be identified early in the development process? This can be re-phrased as what features of the problem domain indicate that a DAS will provide an appropriate solution? There are two non-exclusive classes of environment which imply a need for dynamic adaptability. The first class is where the requirements that are consequent on the environment change on a time-scale likely to be experienced by the running system. For example, a mobile device may need the ability to adapt in order to take advantage of new services as they come in range and become available. The second class is where the trade-offs between non-functional requirements (NFRs) varies with context. Here, the set of requirements may be constant, but what constitutes their satisfaction is not. We hypothesize that the second class is the more common but also more subtle and harder to recognise. For example, in the case of the mobile device above, the choice of service to use may be constrained by a preference for certain service providers that may not always be available.

In the next section we examine two examples of DASs to illustrate that NFR trade-offs are a common feature of each. By so doing they provide evidence in support of our hypothesis.

3 DAS Exemplars

Our first example DAS is an image viewer that adapts to usage patterns and available resources. The system, presented in [4] loads images either from the local file system or a remote URL, and caches images to reduce latency when there is sufficient memory available. Although no requirements process is reported for the system, it is trivial to elicit the two primary non-functional requirements that the adaptation addresses: minimise latency when switching between images and minimise memory usage to avoid swapping. The time taken to load images from local and remote file systems is variable, as is the amount of memory available.

During normal operation, the “minimise memory usage” NFR is given priority, with the system performing no caching. However, when image loading and decoding time exceeds a given threshold, the system adds a caching component, satisficing the “minimise latency” NFR. The viewer also monitors free memory, disabling the cache when scarce and using parametric adaptation to adjust cache size during operation.

Parametric adaptation is also used to switch cache replacement policy: selecting a Most Recently Used policy if images are accessed sequentially, and a Least Recently Used policy otherwise. This essentially tunes the system to best satisfy the “minimise latency” NFR according to usage. The key point is that what constitutes satisfaction of the NFRs varies with the operating environment, thus making adaptation advantageous.

Our second example is an adaptive flood prediction and monitoring system deployed on the banks of the river Ribble in North West England [5]. *GridStix* is an intelligent wireless sensor network that monitors the river and analyses data gathered by multiple sensor nodes. The sensor nodes have enough processing power to process the data co-operatively by acting as a lightweight computational grid, obviating the need to transmit raw water depth and flow rate data off-site for processing. This is significant because *GridStix*’s remote location means that only low-bandwidth

cellular radio networks are available for long-range data transmission. The remote location also means that GridStix is dependent on batteries and solar panels for its power supply. Another feature mandated for GridStix is the use of digital camera images for flow sensing. Digital cameras are inexpensive and robust but produce large volumes of data. The ability to process this data locally is a precondition for the use of digicams.

GridStix's environment has been characterized by domain experts according to three distinct *domains*. In the first the river is *quiescent*. In the second domain, *high flow*, the river flows rapidly but still without significant depth increase. A high flow rate can presage the arrival of a pulse of water that would result in the third domain, *flood*, where both the flow rate and the depth are high. GridStix's key NFRs are "energy efficiency" to maximise battery life, "accuracy" to provide timely and accurate flood warnings, and "fault tolerance" to aid survivability. Crucially, the relative importance of the NFRs varies with the domain. In the quiescent domain, energy efficiency has the priority. With no flood event imminent, the key requirement is to keep the system in readiness, sampling data relatively infrequently. In the high flow domain, the possibility of the onset of a flood event means that accuracy of prediction is relatively more important than it is in the quiescent domain. This means that sampling needs to happen more frequently and the data needs to be processed more quickly. In the flood domain, GridStix still needs to provide accurate predictions but the ability to survive node loss due to submersion or water-borne debris promotes the relative importance of fault-tolerance.

GridStix needs to adapt to the three domains to ensure the appropriate trade-offs between the three NFRs. A reflective middleware platform supports this by, for example, substituting components for different spanning tree algorithms that enable the sensor nodes to communicate. A relatively energy-efficient shortest-path algorithm is used for the quiescent and high flow domains. A more power-hungry but resilient fewest-hop algorithm is used for the flood domain.

Many flood warning systems use sensor networks. Most of these are 'dumb', with no grid-like computational capability. This precludes, for example the use of inexpensive digital camera imaging for flow sensing since the volumes of data are too high to transmit off-site for processing over low-bandwidth communication networks. Nevertheless, such systems are subject to many of the same NFRs as GridStix. Satisfaction of both the fault-tolerance and energy-efficiency requirements is significantly inhibited, however, if the systems are unable to adapt as their river environments change. Hence, although flood warning systems need not necessarily be DASs, the peculiar combination of NFRs to which they are subject make a strong case for them being implemented as DASs. The same argument can be made in many other domains where dynamic adaptability offers better solutions than have hitherto been available.

Both our exemplars exhibit environment volatility. The image processing system has to cope with network latency, while the flood warning system has to cope with a river subject to frequent heavy rainfall. In both cases, the key goals of the system remain the same irrespective of the environment; to render images and to predict flooding, respectively. In both cases, however, the acceptable trade off between their NFRs varies. We hypothesise that this NFR trade-off characteristic is a key signifier that dynamic adaptation is needed.

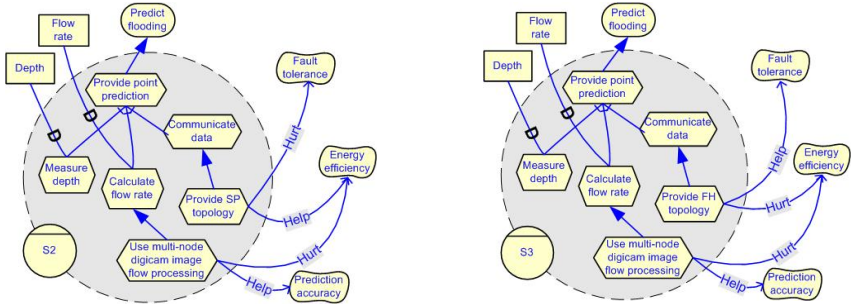


Fig. 1. Models of GridStix configured for *High Flow* (S2) and *Flood* (S3) domains

In [10] we have proposed the use of i^* [14] for making the trade-offs between NFRs explicit. Figure 1 illustrates this by showing developments of two models of how GridStix is configured for the High Flow and Flood domains. The key features are the three softgoals representing the NFRs on the right of each part of the figure. Notice how Fault tolerance and Energy efficiency are either helped or hurt by substituting the task *Provide FH (fewest hop) Topology* for the *Provide SP (shortest path) Topology* as the system adapts from High Flow to Flood. In our approach, which follows closely the three levels of RE for DASs proposed by Berry et al. [9], the models in Figure 1 are developed following development of a strategic dependency graph that models in which the overall goals and softgoals are identified. Subsequent models are developed to specify the adaptation scenarios and to inform the selection of the adaptive infrastructure.

4 Conclusion

Dynamic adaptation allows us to create systems able to operate in environments that have hitherto posed daunting problems for system developers. As ubiquitous computing begins to demand greater context-awareness and flexibility we will encounter problem domains requiring dynamic adaptation increasingly often. Since adaptive systems are fundamentally more complex than static systems, however, being able to identify such problems early on in the RE process is important.

There currently exists no systematic means to recognize the characteristics of a problem that requires a dynamically adaptive solution. Great advances have been made in the development of adaptive infrastructures but the RE community has been slow to respond to the challenges posed by the kinds of problem that adaptive infrastructures are designed to support. The RE community is now beginning to show some awareness, as evidenced by, for example [2] [9] [11].

Our aim in writing this paper has been to argue that a key capability of RE is early recognition of whether a problem demands a dynamically adaptive solution. We have not shown that this can be done in all cases. Rather, we have posited the idea that where analysis of a problem identifies a set of NFRs whose relative priorities change according to the state of the environment, a capability for dynamic adaptability *may*

be a key requirement of the solution. Two exemplars have illustrated our ideas. We now need to test our hypothesis in a wider range of applications to see whether our hypothesis holds. If it does hold, then we will have a useful litmus test of one aspect of complexity that impacts significantly on system development.

References

1. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. *IEEE Computer* 36(1) (2003)
2. Fickas, S.: Clinical requirements engineering. In: *Proceedings of the 27th International Conference on Software engineering* (2005)
3. Cerpa, A., Estrin, D.: ASCENT: adaptive self-configuring sensor networks topologies. *Transactions on Mobile Computing* 3(3) (2004)
4. David, P.C., Ledoux, T.: Towards a Framework for Self-Adaptive Component-Based Applications. In: Stefani, J.-B., Demeure, I., Hagimont, D. (eds.) *DAIS 2003*. LNCS, vol. 2893, pp. 1–14. Springer, Heidelberg (2003)
5. Hughes, D., Greenwood, P., Coulson, G., Blair, G.: GridStix: supporting flood prediction using embedded hardware and next generation grid middleware. *World of Wireless, Mobile and Multimedia Networks* (2006)
6. McKinley, P.K., Sadjadi, S.M., Kasten, E.P., Cheng, B.H.C.: Composing adaptive software. *IEEE Computer* 37(7) (2004)
7. Kramer, J., Magee, J.: *Self-Managed Systems: an Architectural Challenge*. *Future of Software Engineering* (2007)
8. Karsai, G., Ledeczi, A., Sztipanovits, J., Peceli, G., Simon, G., Kovacsashy, T.: An Approach to Self-adaptive Software Based on Supervisory Control. In: Laddaga, R., Shrobe, H.E., Robertson, P. (eds.) *IWSAS 2001*. LNCS, vol. 2614, pp. 24–38. Springer, Heidelberg (2003)
9. Berry, D.M., Cheng, B.H., Zhang, J.: The four levels of requirements engineering for and in dynamic adaptive systems. In: *Proc. 11th International Workshop on Requirements Engineering: Foundation for Software Quality*, Porto, Portugal (2005)
10. Sawyer, P., Bencomo, N., Hughes, D., Grace, P., Goldsby, H., Cheng, B.: Visualizing the Analysis of Dynamically Adaptive Systems Using i* and DSLs. In: *Proc. 2nd Intl. Workshop on Requirements Engineering Visualization*, Delhi, India (2007)
11. Goldsby, H., Cheng, B.H.C.: Goal-Oriented Modeling of Requirements Engineering for Dynamically Adaptive System. In: *Proc. 14th IEEE International Requirements Engineering Conference*, Minneapolis, USA (2006)
12. Sora, I., Cretu, V., Verbaeten, P., Berbers, Y.: Managing Variability of Self-customizable Systems through Composable Components. *Software Process: Improvement and Practice* 10(1) (2005)
13. Efstratiou, C., Cheverst, K., Davies, N., Friday, A.: An Architecture for the Effective Support of Adaptive Context-Aware Applications. In: *Proc. Second International Conference on Mobile Data Management*, Hong Kong (2001)
14. Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In: *3rd IEEE Int. Symp. on Requirements Engineering (RE 1997)*, Washington D.C., USA (1997)

Author Index

- Alenljung, Beatrice 52
Aranda, Jorge 153
- Ballejos, Luciana C. 73
Bendjenna, Hakim 23
Brinkkemper, Sjaak 6
- Charrel, Pierre-Jean 23
- Daneva, Maya 147
- Easterbrook, Steve 153
- Finkelstein, Anthony 88
Fricker, Samuel 37
- Gonnet, Silvio M. 73
Gotel, Orlena C.Z. 129
Grünbacher, Paul 37, 183
Graf, Florian 183
- Harman, Mark 88
Horkoff, Jennifer 153
Howells-Morris, Rhydian 168
- Jones, Sara 109
- Kaschek, Roland 135
Kop, Christian 135
- Lockerbie, James 58
- Mader, Angelika 141
Maiden, Neil 58, 109, 168, 183
Marinčić, Jelena 141
Mayr, Heinrich C. 135
- Montagna, Jorge M. 73
Morris, Stephen J. 129
- Ncube, Cornelius 58
- Paech, Barbara 1
Persson, Anne 52
- Regnell, Björn 123
Rolland, Colette 1
- Saeki, Motoshi 6
Sandkuhl, Kurt 95
Sawyer, Pete 198
Schlosser, Claudia 109
Seyff, Norbert 183
Shekhovtsov, Vladimir A. 135
Strohmaier, Markus 153
Svensson, Richard Berntsson 123
- Thörn, Christer 95
- van de Weerd, Inge 6
Versendaal, Johan 6
- Webers, Wolfram 95
Welsh, Kristopher 198
Wieringa, Roel 141
Wnuk, Krzysztof 123
- Yu, Eric 153
- Zachos, Konstantinos 168
Zarour, Nacereddine 23
Zhang, Yuanyuan 88